ZS01/1

# THE ENGLISH ELECTRIC CO., LTD.

Tel. 700 NELSON RESEARCH LABORATORIES STAFFORD

Report No. NS y 86

Date 17.1.58.

Reference

Order No. Bristol

Report by Aero Engines.

DEUCE Programme News - No. 19. January, 1958.

Tabular Interpretive Scheme. (Version I).

Front Sheet.

Data Sheets 1 - 17.


SUMMARY.

The attached report has been devised and prepared by Bristol Aero Engines Ltd. and describes the tabular interpretive programme. This has been designed to carry out on DEUCE the kind of calculation frequently done by hand on sheets of paper, tabulated in columns where an arithmetic or table look up operation is performed on a column of figures and the results written in another column.

The report is written specifically for those with no experience of computers.

It is intended in due course to replace this scheme by a more powerful and flexible scheme.


MATHEMATICS DEPARTMENT.

HEF

Z.501/23 NELSON RESEARCH LABORATORIES
STAFFORD E. E. CO. LTD.
MATHEMATICS DEPARTMENT.

Continuation to: NS y 86
Sheet No.: 1.

# DEUCE PROGRAMME NEWS - No. 19. January, 1958.

## Tabular Interpretive Scheme (Version I)

CONTENTS.


1. Introduction.

2. Arithmetic in Columns.

3. Dispensing with Tables.

4. Modifying Instructions.

5. DEUCE Operating Instructions.

6. Summary of Codewords.

## 1. INTRODUCTION.

It is well known that programming problems for DEUCE is a specialised and sometimes lengthy procedure. In the past this limited application of the computer mainly to work of a repetitive nature where the initial cost of programming could be offset by the programme's subsequent usefulness. In view of this, a large quantity of important but non-repetitive work remained outside the scope of the computer. The concept and introduction of interpretive schemes has largely removed this difficulty in certain classes of work.

The basic difference between a normal programme and an interpretive programme can be very simply stated. In a normal programme each instruction directs a single operation, whereas an interpretive programme uses master instructions to trigger off a whole set sequence of operations. These master instructions, which are referred to as codes, will obviously be fewer in number than the total of instructions in a programme and can therefore be more quickly compiled. Moreover since these codes have to be interpreted (de-coded) by the programme itself before they can be obeyed, some simple notation can be employed which need have nothing whatever to do with the normal machine language.

The scheme described here is complimentary to the General Interpretive Programme, and to Alphacode, there being different fields of application in which each of the three interpretive schemes in turn have particular value.

One requirement which has to be met is that all problems must be presented in some standard manner, and it is felt that this is best achieved by the system of column tabulation universally used with desk calculating machines. This has the advantage of flexibility and familiarity. Any engineer capable of presenting a calculation to desk machine operators in this form will find no difficulty in using the interpretive scheme since the process is exactly the same.

Although some degree of versatility is achieved in this scheme it is not entirely comprehensive nor is it always desirable to use it in preference to special programmes made for specific problems. The first and most obvious limitation is that it can only deal with the class of calculation capable of being carried out by standard tabular methods and then only when the specified operations are available in the interpretive programme. The second limitation is on machine time. The interpretive scheme is slower in operation than normal DEUCE programmes and therefore where a calculation is of a type frequently repeated machine time can be saved by writing a conventional programme. Thirdly if the amount of data is large compared with the quantity of arithmetic involved the calculation would be better undertaken on a desk machine. This is not peculiar to the interpretive scheme but to digital computers in general since in these cases punching data on to cards may be almost as lengthy as carrying out the whole calculation on a desk machine.

The examples given in this report are meant to serve only for purposes of illustration, they are in no way typical since the amount of arithmetic involved in them is relatively trivial and the use of DEUCE for calculations of this size could probably not be justified.

This scheme should not be used for calculations involving matrix operations. The existing matrix interpretive scheme is far more economical on machine time for problems of this type.

The design of the interpretive scheme makes allowance for the inclusion of more operations than those given in this report. Only experience in using the scheme will indicate which additional operations will be most useful. For this reason some blanks have been left for the present, to be filled later as experience dictates.

## 2. ARITHMETIC IN COLUMNS.

### 2.1 General Tabular Method.

Given a sheet of paper divided into columns and rows with initial data in the first few columns calculation normally proceeds on a desk machine by taking one or two columns of numbers. carrying out some arithmetical operation on these numbers and putting the result in a third column, the tabulation carrying on in this way as the solution is built up. Constants such as $\pi$ , Young's Modulus, expansion indices and so on are not usually given a column each but are written down separately.

The requirements of the interpretive scheme are exactly the same, except that the sheet of paper can be assumed to exist inside DEUCE and that DEUCE will do all the calculation and filling in of columns on being told precisely what to do. This "sheet of paper" is also divided into columns and rows, 128 columns (numbered 0 to 127) and 32 rows (numbered 0-31). The first column (column 0) is reserved for storing constants. Because of this the rows of column 0 are not related in any way to the rows on the remainder of the sheet. The rows of column 0 are numbered $C_0$ $C_1$ $C_2$ $C_3$ etc.

This layout is illustrated below:-

| | 0 | | 1 | 2 | 3 | | n | | 126 | 127 |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | |
| $C_0$ | | | | | | | | | | |
| $C_1$ | | | | | | | | | | |
| $C_2$ | | | | | | | | | | |
| $C_3$ | | | | | | | | | | |
| | | | | | | | | | | |
| $C_{31}$ | | | | | | | | | | |

When doing a tabulation on a desk machine set each set of results is usually written down in the next vacant column. With the interpretive scheme new results can be written over columns previously obtained but no longer required in the calculation. No special instructions for this are required, the act of "writing" in any column automatically deletes anything previously in that column. The economy on space achieved in this way allows large tabulations to be handled which would otherwise require more than the 127 columns available. Since, in general, the answers to any particular operation are not put in the next vacant column it is necessary to specify the column for the results. This information will be contained in the codewords.

## 2.2 Codewords

Codewords represent an abbreviated form of instruction for carrying out some operation in tabular arithmetic. One codeword contains enough information to define an operation, thus these codewords are equivalent to the instructions which would be given to a desk machine operator to do a similar job. The interpretive programme takes the codewords in sequence, decodes them, selects the columns of data on which the specified operations are to be performed, brings into action the appropriate programme and plants the results in the required columns.

Codewords are stored in four separate blocks, 31 in block 1, and 32 in each of the remaining three blocks and are numbered consecutively from 1 to 127. Sets of up to 127 codewords are read in and stored in the appropriate blocks automatically. Should a calculation require more than 127 codewords an instruction can be given to read in a new block. Details of how this is done are described in Section 3.

With one exception codewords consist of four numbers. The first two are the numbers of the columns on which the operation (multiplication, addition etc.) is to be performed. The third number gives the number of the column to which the result is to be sent, and the fourth number represents, in code, the operation which is to be performed. Thus, assuming the code number for division to be 1, the codeword

3 2 5 1

would be interpreted as:-

divide the numbers in column 3 by the corresponding numbers in column 2 and put the results in column 5.

## 2.3 Column Arithmetic.

The codewords for the basic arithmetic operations, reading and printing are:-

| Codeword | | | | Interpretation |
|---|---|---|---|---|
| a | b | c | 0 | (Col. a) x (Col. b) result in (Col. c) |
| a | b | c | 1 | (Col. a) ÷ (Col. b) result in (Col. c) |
| a | b | c | 2 | (Col. a) + (Col. b) result in (Col. c) |
| a | b | c | 3 | (Col. a) - (Col. b) result in (Col. c) |
| 0 | 0 | c | 4 | Read column of data from cards and write in column c. |
| a | 0 | 0 | 5 | Print out column a. |
| a | 0 | c | 6 | Find the square root of column a and put the result in column c. |

Throughout the scheme the results of all arithmetic operations are given to six significant figures, the last of which may have an error of 1.

Example:-

Calculate a table of second moments of area of hollow tubular shaft sections for a fixed inside diameter of 4 inches and outside diameters ranging from 4.10 to 4.50 inches in steps of 0.05 inches, for which the formula is:-

$$I_n = \frac{\pi}{64} (D_n^4 - d_n^4)$$

Since D is numerically close to d the formula is better written for computing purposes as:-

$$I_n = \frac{\pi}{64} (D_n - d_n) (D_n + d_n) (D_n^2 + d_n^2)$$

Putting this in tabular form ready for evaluation on a desk machine might result in a layout such as the one given below.

| | 0 | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | $\frac{\pi}{64}$ | d | D | $D^2$ (3) x (3) | $d^2$ (2) x (2) | $D^2+d^2$ (4) + (5) | D + d (3) + (2) | D - d (3) - (2) | (D - d) x(D+d) (7) x (8) | $D^4-d^4$ (9) x (6) | I (1) x (10) |
| $C_0$ | | | .049087 | 4.0 | 4.10 | | | | | | | | |
| $C_1$ | | | .049087 | 4.0 | 4.15 | | | | | | | | |
| $C_2$ | | | .049087 | 4.0 | 4.20 | | | | | | | | |
| | | | | | | | etc | | | | | | |
| $C_n$ | | | .049087 | 4.0 | 4.50 | | | | | | | | |

To work this example by means of the interpretive scheme, the first three columns of data would be required together with the following set of codewords which replace the "operation" row in the above table.

| Codeword | | | | | What codeword does |
|---|---|---|---|---|---|
| (1) | 0 | 0 | 1 | 4 | Read data into Col. 1. |
| (2) | 0 | 0 | 2 | 4 | Read data into Col. 2. |
| (3) | 0 | 0 | 3 | 4 | Pead data into Col. 3. |
| (4) | 3 | 3 | 4 | 0 | (Col.3) x (Col.3) putting result in (Col.4) |
| (5) | 2 | 2 | 5 | 0 | (Col.2) x (Col.2) putting result in (Col.5) |
| (6) | 4 | 5 | 6 | 2 | (Col.4) + (Col.5) putting result in (Col.6) |
| (7) | 3 | 2 | 7 | 2 | (Col.3) + (Col.2) putting result in (Col.7) |
| (8) | 3 | 2 | 8 | 3 | (Col.3) - (Col.2) putting result in (Col.8) |
| (9) | 7 | 8 | 9 | 0 | (Col.7) x (Col.8) putting result in (Col.9) |
| (10) | 9 | 6 | 10 | 0 | (Col.9) x (Col.6) putting result in (Col.10) |
| (11) | 1 | 10 | 11 | 0 | (Col.1) x (Col.10)putting result in (Col.11) |
| (12) | 11 | 0 | 0 | 5 | Print out (Col.11) |
| (13) | 0 | 0 | 0 | 32 | Stop. |

When the calculation is complete and the required columns have been printed out the final codeword should be stop instruction as shown above.

## 2.4 Constants.

In the example just given "$\frac{\pi}{64}$" and "d" remain constant from one row to the next. Each required a whole column when in fact one space would have been sufficient. Clearly, if a large number of these constant terms had been involved, much waste of space would have resulted. To overcome this disadvantage column 0 is reserved for constants as previously mentioned. The rows of column 0 are numbered $C_0$ $C_1$ $C_2$ etc. To operate on a column and a constant, the operation number of the codeword must be 7. The code r denoting the actual operation, multiplication, a dition etc., is placed in brackets in the centre of the codeword. Hence when constants are involved, the codeword takes on the slightly modified form.

a n (r) c 7

a and c have the same significance as before but b (the number of the second column to be operated on) no longer applies and is replaced by n, where n is the number of the row in Col. 0 containing the required constant $C_n$. Four different codewords of this form are possible:-

a n(0) c 7 $\qquad$ $C_n$ x (Col.a) answer in Col. c.

a n(1) c 7 $\qquad$ $C_n \div$ (Col.a) answer in Col. c.

(Note the operation (Col.a) $\div C_n$ is not required since $\frac{1}{C_n}$ can be given in place of $C_n$ initially.

a n(2) c 7 $\qquad$ $C_n$ + (Col.a) answer in Col. c.

a n(3) c 7 $\qquad$ $C_n$ - (Col.a) answer in Col. c.

(Note (Col.a) - $C_n$ operation is never required. It can be obtained by (Col. a) + $(-C_n)$)

A special codeword is required to read in a set of constants from cards into column 0:-

p 0 0 4 $\qquad$ Read p constants from data cards into Col. 0.

The example just given can be used to illustrate use of these codewords. The layout can be expressed as follows:-

| | 0 | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | D | $D^2$<br>1 x 1 | $D^2+d^2$<br>2 + $C_2$ | D + d<br>1 + $C_1$ | D - d<br>1 + $C_3$ | 3 x 4 | 6 x 5 | 7 x $C_0$ | |
| $C_0$ | .049087 | $= \frac{\pi}{64}$ | 4.10 | | | | | | | | |
| $C_1$ | 4.0 | = d | 4.15 | | | | | | | | |
| $C_2$ | 16.0 | $= d^2$ | 4.20 | | | | | | | | |
| $C_3$ | -4.0 | = -d | | | | | | | | | |

The codewords in this case would be:-

| | | | | | |
|---|---|---|---|---|---|
| (1) | 4 | 0 | 0 | 4 | Read 4 constants into Col. 0 |
| (2) | 0 | 0 | 1 | 4 | Read D's into Col. 1. |
| (3) | 1 | 1 | 2 | 0 | (Col.1) x (Col.1) put in Col. 2. |
| (4) | 2 | 2(2) | 3 | 7 | (Col.2) + $C_2$ put in Col. 3. |
| (5) | 1 | 1(2) | 4 | 7 | (Col.1) + $C_1$ put in Col. 4. |
| (6) | 1 | 3(2) | 5 | 7 | (Col.1) + $C_3$ put in Col. 5. |
| (7) | 3 | 4 | 6 | 0 | (Col.3) x (Col.4) put in Col. 6. |
| (8) | 6 | 5 | 7 | 0 | (Col.6) x (Col.5) put in Col. 7. |
| (9) | 7 | 0(0) | 8 | 7 | (Col.7) x $C_0$ put in Col. 8. |
| (10) | 8 | 0 | 0 | 5 | Print out Col. 8. |
| (11) | 0 | 0 | 0 | 32 | Stop. |

### 2.5 Storage Economy.

Remembering that unwanted columns can be written over, the amount of space required could have been considerably reduced as shown by the following set of codewords which does the same job using only 3 columns (excluding Col. 0).

| | | | | | | |
|---|---|---|---|---|---|---|
| (1) | 4 | 0 | 0 | 4 | Read in constants into Col. 0. | |
| (2) | 0 | 0 | 1 | 4 | Read D's into Col. 1. | |
| (3) | 1 | 1 | 2 | 0 | (Col.1) x (Col.1) in Col.2. | ($D^2$ in Col.2) |
| (4) | 2 | 2(2) | 2 | 7 | (Col.2) + $C_2$ in Col. 2. | ($D^2+d^2$ in Col.2) |
| (5) | 1 | 1(2) | 3 | 7 | (Col.1) + $C_1$ in Col. 3. | (D+d in Col.3) |
| (6) | 2 | 3 | 2 | 0 | (Col.2) x (Col.3)in Col. 2. | $(D^2+d^2)(D+d)$ in Col.2). |
| (7) | 1 | 3(2) | 3 | 7 | (Col.1) + $C_3$ in Col. 3 | (D-d in Col.3) |
| (8) | 2 | 3 | 3 | 0 | (Col.2) x (Col.3) in Col.3 | ($(D^2+d^2)(D+d)(D-d)$ in Col.3). |
| (9) | 3 | 0(0) | 3 | 7 | (Col.3) x $C_0$ in Col.3 | ($\frac{\pi}{64}(D^4-d^4)$ in Col.3) |
| (10) | 3 | 0 | 0 | 5 | Print out column 3. | |
| (11) | 0 | 0 | 0 | 32 | Stop. | |

## 3. DISPENSING WITH TABLES.

It is not normally convenient to store tables of functions in digital computers. All the more commonly used functions can be calculated more quickly in the machine than they could be read from stored tables. It is for this reason that the interpretive scheme contains sub-programmes to calculate the more frequently used functions.

### 3.1 Trigonometrical Functions.

Given a column of angles (in radians) in Col. a, the sines of these angles are put in column C by the codeword:-

a 0 c 11

and the cosines by:-

a 0 c 12

No facility is provided for tangents since these can be readily evaluated from the sines and cosines.

The inverse process, obtaining angles (in radians) from given trigonometrical ratios may also be found directly. Given sine x in column a, x is put in column c by the codeword:-

a 0 c 20

similarly the codeword for $\cos^{-1}x$ is:-

a 0 c 21

3.2 Exponential and Logarithmic Functions

To calculate log x where x is in column a and log x is required in column c, the codeword is:-

a 0 c 8

The anti-log of p (or $e^p$), where p is in column a and $e^p$ is required in column c, is obtained with the codeword:-

a 0 c 9

Example: Suppose that in the course of a calculation, the variables $P_1$ $P_2$ and $\gamma$ have been evaluated and put in columns 15, 16 and 17 respectively, and that the number -1, appears in $C_8$, and that it is now necessary to evaluate the expression:-

$$y = \left(\frac{P_1}{P_2}\right)^{\frac{\gamma - 1}{\gamma}}$$

The following procedure could be used:-

| | | | | |
|---|---|---|---|---|
| 16 | 16 | 18 | 1 | $\frac{P_1}{P_2}$ in col. 18 |
| 17 | 8(2) | 19 | 7 | $\gamma - 1$ in col. 19. |
| 19 | 17 | 19 | 1 | $\frac{\gamma - 1}{\gamma}$ in col. 19. |
| 18 | 0 | 18 | 8 | $\log\left(\frac{P_1}{P_2}\right)$ in col. 18. |
| 18 | 19 | 18 | 0 | $\left(\log\frac{P_1}{P_2}\right)\left(\frac{\gamma - 1}{\gamma}\right)$ in col. 18. |
| 18 | 0 | 18 | 9 | Anti-log = y in col. 18. |

The required result is in column 18, column 19 at the end of this sequence contains $\frac{\gamma - 1}{\gamma}$ and can be written over if not required. The original columns containing $P_1$ $P_2$ and $\gamma$ have been left undisturbed.

3.3 Modulus and Shift.

As will be seen in the later section on discrimination it is sometimes convenient to form the modulus of a column of numbers. The codeword:-

a 0 c 14

will do this, putting $|x|$ in col. c if x is in column a.

A further useful operation is the column shift.

The codeword:-

a 0 c 10

will produce in column c the contents of column a shifted down one place. Finite difference calculus frequency employs differences in the form $X_n - X_{n-1}$. This can be done with two codewords. As an example of this suppose a column of x's ($X_0$ $X_1$ $X_2$ ... ) are in column 4 and it is required to calculate $X_n - X_{n-1}$ in column 6. The two instructions necessary would be:-

4 0 5 10 Put shifted down version of col. 4 in col. 5.

4 5 6 3 Subtract column 5 from column 4 and put in Col. 6.

After these two instructions, columns 4, 5 and 6 would appear as:-

| 4 | 5 | 6 |
|---|---|---|
| $X_0$ | 1 | $X_0 - 1$ |
| $X_1$ | $X_0$ | $X_1 - X_0$ |
| $X_2$ | $X_1$ | $X_2 - X_1$ |
| | | |
| $X_n$ | $X_{n-1}$ | $X_n - X_{n-1}$ |

It will be observed that after a shift the number 1.0 is written (automatically) in row 0 of the shifted column n. Had this not been arranged, any attempt to divide by that column would have entailed trying to divide by the zero in row 0. Since row 0 now contains an arbitrary number any further operations involving this column will invalidate the corresponding elements in row 0. This is inevitable if differencing is done, but is of no consequence since after differencing the computation always proceeds with one row less. Further reference is made to this in the following section.

3.4 Summation.

If column a contains $X_0$ $X_1$ $X_2$ ... $X_n$ ...

$\sum_0^n X_n$ is formed in column c by the instruction

a 0 c 19

After this instruction columns a and c appear as:-

| a | c |
|---|---|
| $X_0$ | $X_0$ |
| $X_1$ | $X_0 + X_1$ |
| $X_2$ | $X_0 + X_1 + X_2$ |
| | |
| $X_n$ | $\sum_0^n X_n$ |

This facility can be used to perform first order numerical integration.

Suppose it is required to evaluate $y_n = \int_{X_0}^{X_n} F(X)\,dx$

If the values of $F_n$ (n) appear in, say, column 12 and $X_n$ columns 13, then $y_n$ can be readily obtained. It is first necessary to have in one of the columns the configuration:-

| |
|---|
| 0 |
| 1 |
| 1 |
| 1 |

etc.

If column a is multiplied by this column, the result will be the same as the original column "a" with the first number zero. This is a necessary preliminary to summation if a shift operation has been previously performed. After a shift operation row 0 becomes invalid (as shown in section 2.3), but anything in row 0 of a particular column will be included if a summation is made on that column. It will be assumed for this particular illustration that the above configuration appears in column 100.

Expressing the integral in finite difference form:-

$$y_n = \sum_0^n \left(\frac{F_n(X) + F_{n-1}(X)}{2}\right)(X_n - X_{n-1})$$

the codewords to do this would be:-

| | | | | |
|---|---|---|---|---|
| 12 | 0 | 14 | 10 | Form $F_n(X) + F_{n-1}(X)$ in col.14. |
| 12 | 14 | 14 | 2 | |
| 14 | 5(0) | 14 | 7 | Form $\frac{1}{2}$ $(F_n(X) + F_{n-1}(X))$ in col.14.($C_5=\frac{1}{2}$) |
| 13 | 0 | 15 | 10 | Form $(X_n - X_{n-1})$ in Col.15. |
| 13 | 15 | 15 | 3 | |
| 14 | 15 | 14 | 0 | Col.14 x Col.15 in Col.14. |
| 14 | 100 | 14 | 0 | Put zero in first row of Col.14 (see above) |
| 14 | 0 | 14 | 19 | $y_n$ in Col.14. |

3.5 Storing Tabulated Number.

The interpretive scheme is designed to operate on columns of numbers and no facility is provided for working with individual numbers except in the case of the constants stored in column 0.

Occasionally however, an operation with a single number may be unavoidable. As an example, if a column has been summed by means of the summation operation, the column total (a single number) may be required in some other part of the calculation. In order to make this possible a facility has been made to take any single number from any column and store it in one of the positions in column 0. From here it is available for operations in the manner already described for constants. The codeword:-

a b c 13

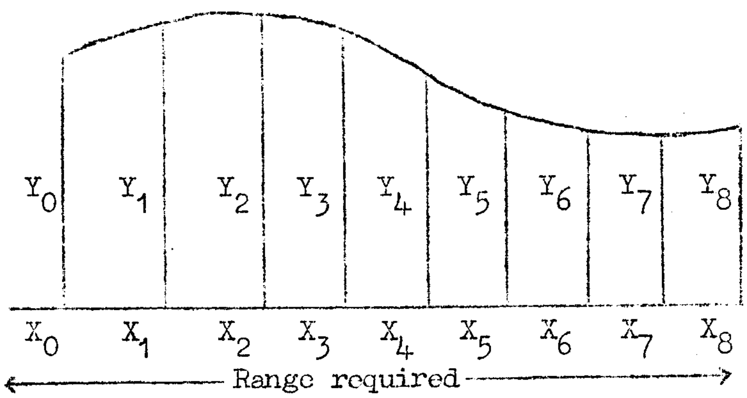
is interpreted in this instance as:-

Take the number in Col. a row b and store in position $C_c$ (Note rows are numbered 0-31).

3.6 Graphical Data.

Many empirical functions are represented in graphical form and as such cannot be presented to a digital computer. Methods are available for obtaining values of these functions from a set of given co-ordinates. No one method, however, will do all types of curves and the normal practice in writing conventional DEUCE programmes is to select curve fitting methods appropriate to the graphical data supplied, even though this may mean using more than one method in the same programme. The method selected for the interpretive scheme uses second order interpolation on a set of values of a function given at equal intervals of the argument, and for functions of a single variable only. The "graph" must be presented as a column of numerical data in this manner:-

Col. 6.
$\Delta x$
$X_0$
$Y_0$
$Y_1$
$Y_2$
etc.
$Y_{n-1}$




$X_0$ is the lowest value of X and $\Delta X = X_n - X_{n-1}$ (constant for any one graph). One y value more than the number required to cover the range must be given as indicated by the dotted extrapolation in the above figure. The maximum number of points that can be used to specify a curve is 30 (this number plus $\Delta X$ and $X_0$ occupying the whole of column b). The codeword (n+2) 0 c 4 reads the column of graphical data into column c, n being the number of points specifying the curve.

The codeword:-

a b c 15

will take values of X from column a "read and graph" given by column b and put the results (y's) in column c. This programme will not extrapolate.

e

3.7 Graphical Data - Linear Interpolation.

Before using second order interpolation in curve fitting some care must be taken to ensure the validity of the assumption made, i.e. that an equation of second degree expresses the relation between dependent and independent variables with sufficient accuracy over the selected internal. Whether a second degree equation represents a good fit or not will not be immediately apparent.

To simplify matters a little a linear interpolation programme has been incorporated. This will interpolate over a range specified by up to fifteen pairs of co-ordinates which need not be given at equal increments. The choice of co-ordinate distribution will largely determine the fit but since this amounts only to approximating to the curve by a series of straight lines, the magnitude of the errors in fit will be obvious and discontinuities can be followed. For this reason it is recommended that linear interpolation be used to represent graphical data wherever possible.

This programme is additional and not alternative to the second order programme. It is called into operation by the codeword a b c 22. where a contains the column of x's and b the co-ordinates in the following form:-

$$\begin{array}{l} n \\ X_0 \\ Y_0 \\ X_1 \\ \vdots \\ X_{n-1} \\ Y_{n-1} \end{array}$$

n being the number of co-ordinate pairs ($n \leqslant 15$)
The instruction for reading linear graphical data into column c will be

$(2n + 1)$ 0 c 4

## 4. AUTOMATIC MODIFICATION OF INSTRUCTION SEQUENCE.

By using facilities to modify instructions, a considerable gain in programming flexibility can be obtained. As will be seen later the benefits available largely depend on the ingenuity of the individual setting out the tabulation. Once the technique of instruction modification has been understood many applications will become apparent. It is hoped that the examples given here will suffice as a pointer to the possibilities.

### 4.1 Jump to Prescribed Codeword.

Should it be required to break away from the given sequence of codewords, the instruction:-

0 0 p 33

will do no arithmetical operation, but will take as the next instruction codeword p (codewords are numbered 1 to 127), and continue in sequence from that point.

Example:- Suppose a table of functions is required $y_n = F_n(X)$ for values of X from 1 to 100 in steps of 1.0. It will be observed that this would require a tabulation of 100 rows. Since the interpretive scheme is limited to 32 rows this would normally mean calculating $y_n$ for a range of X from 1 to 30 (say) and then repeating with ranges of 31 to 60 etc. Each calculation would consist of the same set of instructions, the only alteration would be the column of X's. (say in Col. 5). If however, the print out instruction were to be followed by the two codewords:-

0 0 5 4 Read fresh set of data into Col. 5.

0 0 p 33 Jump to codeword p.
(Codeword p being the first in the sequence of calculation).

then a new set of X values (from 31 to 60) would replace the original set and the calculation would then begin again. The only precaution necessary would be ensuring that the data (i.e. the data columns) were presented in the correct order. This is easily arranged by presenting the columns of data on the data sheet in the order in which they will be required by the machine. This process can be conveniently illustrated with a block diagram:-

| | | | | |
|---|---|---|---|---|
| (1) | Read in X's | | (1) | Read in nth set of X's |
| (2) | Calculate Y's | | (2) | Calculate Y's. |
| (3) | Print Y's | or | (3) | Print Y's. |
| (4) | Read in more X's | | (4) | Jump to (1) |
| (5) | Jump to (2). | | | |

The number of rows in each calculation must remain in same (the number of rows in any interpretive scheme calculation is fed into DEUCE by an operator at the outset). In the example given, each calculation would consist of 30 rows, and in order to cover the required range, 120 would have to be found. It would in this instance have been more economical on time if 25 rows had been worked in each calculation. The calculation would still have been repeated four times (this would be inevitable since the maximum number of rows is 32) but each would have been done on a reduced number of rows.

4.2 Replacing One Codeword with Another.

In the above example the sequence of instructions being obeyed remained the same each time round the "loop". In some applications one or more instructions will need to be different at each repetition. In these cases some modified versions of the original instructions can be added to the bottom of a list of codewords, these being written over the original codewords by instructions of the form:-

a b c 35

This instruction will do no arithmetic but will cause codeword a to replace codeword c and take as the next instruction codeword b. The instructions will be taken in sequence from codeword b.

4.3 Discrimination.

The discriminating facility enables the machine to take one of three different sets of instructions according to the state of the calculation. Two criteria are used for this purpose. They are:-

(a) Choose on the basis of a prescribed column being all + ve, all -ve or a mixture of both positive and negative.

(b) Choose on the basis of a prescribed column being all zero, all non zero, or a mixture of both.

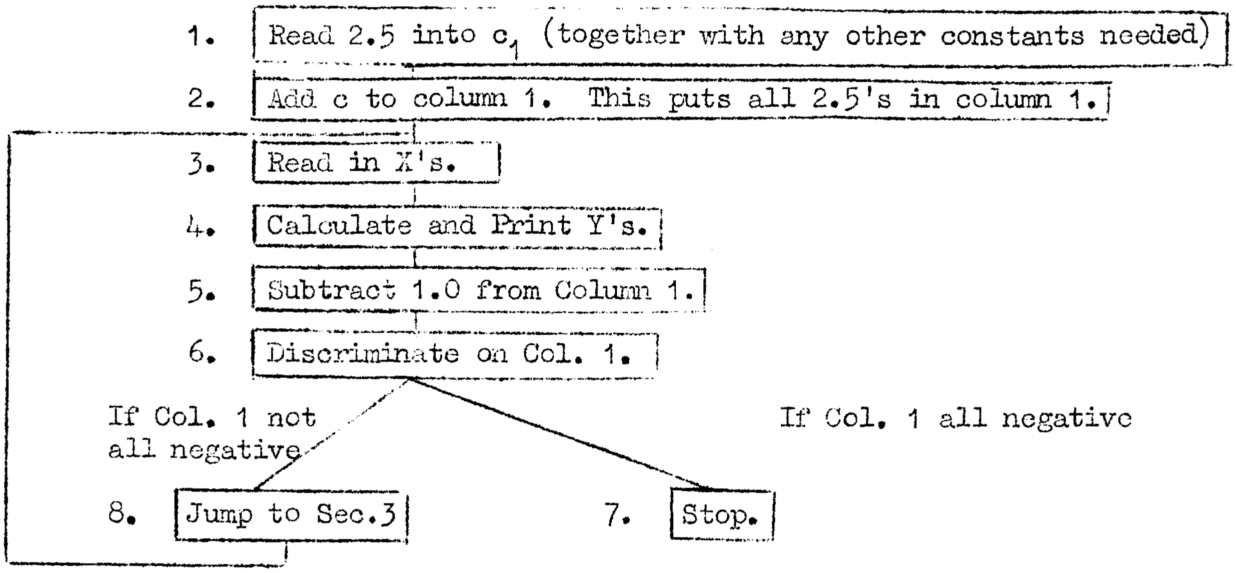
The codeword a b c 16

will do no arithmetic but will cause codeword b to be taken as the next instruction if Col. a is all +ve, codeword c to be taken if Col. a is all -ve, or the next codeword in the sequence if a is neither all +ve, nor all -ve.

The codeword a b c 17

works in exactly the same manner except that a zero or non zero condition replaces the +ve -ve condition.

In the example of section 3.1 there is no means of getting out of the "loop". Were it not for the fact that the programme would run out of data the whole repetitive cycle would go on indefinitely. The block diagram below shows how this state of affairs could have been avoided:-




It will be observed from the discriminating instruction that three alternative paths are available, therefore, if a two-way choice is to be made, as in the above example, two of the paths must be made coincident i.e. must lead to the same instruction.

Section 6 would consist of only one codeword (say the 36th codeword). Section 8, also only one codeword, would have to be the next in sequence (37th codeword). The position of the stop codeword comprising section 7, would be immaterial. Assuming, however, that in this instance the stop codeword is the 50th, then section 6 would be

36) 1 37 50 16

It will be seen from the block diagram that Column 1 is used as a counter. At the end of the first time round Column 1 consists of all 1.5's. On reaching the discrimination instruction the sequence jumps to (8) and so repeats. At the end of the second time round Column 1 contains all .5's hence once again the sequence jumps to (8). At the end of the third time round Column 1 contains all -.5's and is therefore all negative and the discrimination instruction leads on to the 50th codeword, the stop instruction.

It would appear logical to put 3.0 into $c_1$ at the outset and use the zero or non-zero discriminator. In practice, however, this would not be satisfactory since in converting from a decimal to binary some round-off errors inevitably occur and some small remainder may be left where a zero is expected. Should this be so the discriminator will treat this as being non-zero.

4.4 Read in More Codewords.

As was explained earlier codewords are stored in the machine in four blocks numbered 1 to 4, the codeword:-

0 b c 34

will read a fresh block of codewords into block C and take as its next instruction codeword b. This operation obliterates the codewords previously in block C, the serial numbers of the new codewords being the same as the overwritten ones.

## 5. INSTRUCTION TO DEUCE OPERATORS.

### 5.1 Punching of Data.

Data cards are punched in floating decimal $a \times 10^b$ with $1 < a < 10$ in Cols. 2-10 of DEUCE field with sign in Col. 1 b as 9 digit integer in Cols. 12-20 with sign in Col.11.

Cards should be punched and stacked in columns as set out on data sheet.

### 5.2 Punching of Codewords.

Each block of codewords is punched as a triad of cards in the normal fashion but with no initial instructions, the first minor cycle of the first triad should be left blank if more than one triad is to be read in. If only one triad is to be read in, m/c 0 should be filled with the following codeword:-

0 0 1 33

If more than one triad is to be read in (i.e. if m.c.0 of 1st triad is blank,) make up to 4 delay lines with blank cards. Codes are punched in binary as integers x $P_1$; $P_9$; $P_{17}$ and $P_{25}$ the central bracketed code where this appears should be punched as integer x $P_{14}$. A $P_{32}$ punched against any code behaves as a stopper.

For a programme consisting of not more than four D.L.'s of codes the codes will precede the data in stacking order. A programme of more than four D.L.'s will contain an instruction to read more codes and an inspection of the programme will be needed to ascertain the stacking sequence. In front of the code cards a parameter card is required with n $P_1$ on the y row, where n is the number of rows being worked in the tabulation.

### 5.3 Interpretive Programme and Bricks.

Each brick has a fixed position on the drum. The interpretive programme is in two parts, front and back, the bricks being sandwiched between. Only the bricks required for a particular calculation need be assembled into the pack. The front section of the interpretive scheme incorporates "clear drum" and "read to drum". The back section includes A 11 F/1.

### 5.4 Testing Facilities.

A $P_{32}$ on the I.D. will cause the machine to stop on each codeword, displaying the codeword on the O.S. A single shot causes the programme to continue.

To stop on codeword n, put $nP_{17}$ on I.D. single shot to continue. To punch out intermediate column (or to insert an instruction from I.D.) stop machine to appropriate point by $P_{32}$ or $nP_{17}$; T.I.L. on, single shot T.I.L. off, put on stop. Insert required codeword on I.D. (a 0 0 5 if column a is to be punched out), give single shot, clear I.D. put on normal. If the inserted codeword contains a $P_{32}$ the machine will stop with the code displayed on O.S. Give single shot to continue. The next code to be obeyed will be the one due to be obeyed when the I.D. instruction was inserted.

To find out which code is being obeyed when a failure occurs it is only necessary to bring down track 15/7 to D.L.1 on ext. tree and enter in $1_{30}$ with $P_{32}$ on the I.D. The programme will then stop as usual displaying on the O.S. the codeword following that on which the failure occurred.

5.5 Failure Indications.

| | | | | |
|---|---|---|---|---|
| | 3 | 21-29 | x | See A11/1 |
| | 2 | 31-29 | | Divisor zero |
| | | 21-28, 9-24, 21-29 | | Read failure. |
| Square root A | 5 | 27-29 | | A negative. |
| Log A | 6 | 29-29 | | A negative. |
| | 6 | 31-31 | | A zero. |
| Anti log (A (= a x $2^b$)) | 5 | 13-29 | | b $\geqslant$ $2^{12}$ |
| $Sin^{-1}$A or | 6 | 13-29 | x | b > 1 |
| $Cos^{-1}$A | 3 | 14-29 | | b < 1 A > 1.0 |

5.6 Results.

These are punched out column by column, one number per card in standard floating decimal. Since they are in the same form as the data and punched in the same fields they can be used as input if required.

6. SUMMARY OF CODEWORDS.

| CODEWORD | | | | INTERPRETATION. |
|---|---|---|---|---|
| a | b | c | 0 | (Col.a) x (Col.b) putting result in Col.c. |
| a | b | c | 1 | (Col.a) ÷ (Col.b) putting result in Col.c. |
| a | b | c | 2 | (Col.a) + (Col.b) putting result in Col.c. |
| a | b | c | 3 | (Col.a) - (Col.b) putting result in Col.c. |
| 0 | 0 | c | 4 | Read column of data from cards and put in Col.c. |
| (n+2) | 0 | c | 4 | Read graph of n co-ordinates from cards and put in Col.c. |
| p | 0 | 0 | 4 | Read p constants from data cards. |
| a | 0 | 0 | 5 | Print out column a. |
| a | 0 | c | 6 | √ (Col.a) putting result in Col.c. |
| a | n(0) | c | 7 | (Col.a) x $C_n$ putting result in Col.c. |
| a | n(1) | c | 7 | $C_n$ ÷ (Col.a) putting result in Col.c. |
| a | n(2) | c | 7 | (Col.a) + $C_n$ putting result in Col.c. |
| a | n(3) | c | 7 | $C_n$ - (Col.a) putting result in Col.c. |
| a | 0 | c | 8 | log (Col.a) putting result in Col.c. |
| a | 0 | c | 9 | Anti-log (Col.a) or $e^{(Col.a)}$ putting result in Col.c. |
| a | 0 | c | 10 | (Col.a) shifted down one place in Col.c. |
| a | 0 | c | 11 | Sine (Col.a) in Col.c. |
| a | 0 | c | 12 | Cosine (Col.a) in Col.c. |
| a | b | n | 13 | Store number in Col.a row b in position $C_n$. |

| CODEWORD | | | | INTERPRETATION. |
|---|---|---|---|---|
| a | 0 | c | 14 | \|Col. a\| in Col. c. |
| a | b | c | 15 | From x's in Col. a and graph of y = f(x) in Col. b find y's and store in Col. c. |
| a | b | c | 16 | Take as next instruction codeword c if Col. a is all negative or codeword b if Col. a is all positive. Go to next codeword if Col. a contains a mixture of positive and negative numbers. |
| a | b | c | 17 | Take as next instruction codeword c is Col. a contains no zeros, codeword b if Col. a is all zeros or next codeword if Col. a contains both zero and non-zero elements. |
| a | 0 | c | 19 | Given $X_n$ in Col. a form $\sum_{x_0}^{x_n} x_n$ in Col. c. |
| a | 0 | c | 20 | Given sine x in Col. a, forms x in Col. c. |
| a | 0 | c | 21 | Given cosine x in Col. a forms x in Col. c. |
| 0 | 0 | 0 | 32 | Stop programme. |
| 0 | 0 | p | 33 | Jump to codeword p. |
| 0 | b | c | 34 | Read fresh block of codewords into block c and jump to codeword b. |
| a | b | c | 35 | Replace codeword c by codeword a and jump to codeword b. |
| a | b | c | 36 | Replace codeword c by a + b. |