

Deuce Schemes A and B

(developed at N.P.L.)

Contents:

Section I	Deuce Standard Bricks (Schemes A and B)
II	General Interpretive Program (Interim Version)
III	Example of the use of Scheme B programs.
IV	Identification numbers of Scheme A and B programs.
V	Details of the Scheme B programs in the library. (January, 1956).

30th January, 1956



Section I

Deuce Standard Bricks

1. General

Standard Bricks are intended to be assembled together into a single pack of cards capable of performing the desired operation. Programmes are of two types - "initial" and otherwise - and a pack can be assembled from any programmes, provided that the first of these is an initial type.

In order not to interfere with the use of 130 for ordinary 1st order sub-routines, also to fit in with the fact that small programmes usually use DL1 onwards, DL12 will be used for parameters etc., in the same way as DL11 has been used for the Pilot ACE. Thus all 8 N.I.S. DL's are available for instructions, and the parameter DL, DL12, is beyond the magnetic DL, DL11, so that the DL's between the end of the programme and DL11 can be used as a consecutive store.

In more detail, 120 is kept empty, and "initial" programmes will examine 1230, and, if empty, insert 1, 12-24 0, 0. All programmes will end by doing 12-1 (32 mc) and going to 130. TS13 should be left empty at the end. DL input instructions will be based on 10 being empty, not 80, so that DL1 will have to be read in last.

Initial programmes, denoted type (1), have a special initial card which does not need to be removed when the programme occurs in the middle of a pack, and all other programmes, denoted type (2), start with a special card which clears READ and goes to 130. This prevents such programmes being used initially.

With the exception of examining 1230 as above, programmes will leave alone 1216-31 (for possible use by any interpretative schemes that may be devised later), and Col.54 is not used at all, all programmes beginning with either 12-24 or 9-24.

2. Scheme A

The previous paragraphs are general notes which apply to all programmes intended to be used as bricks. The matrix algebra programmes which form Scheme A will now be discussed.

The Matrix Store is defined as starting at 0/00, storing the elements of the  $m \times n$  matrix consecutively without any row sums.  $m \ n \leq 8192$ .

The Diagonal Store is defined as occupying DL40 onwards, with the elements of the order  $N$  diagonal stored in reverse order.  $N \leq 224$  ( $\leq 256$  if mag not used).

The Scalar Store consists of 3 adjacent mc in DL12.

Contrary to PILOT MODEL practice, the B.P. values for Matrix, Diag. and Scalar are stored separately from each other, only being added together in the case of a multiplication of one quantity by another. To avoid any errors due to B.P. values being lost from store, the sum checks for Matrix, Diag. and Scalar all have B.P. added in.

This diagram shows the arrangement. R is not preserved longer than needed, and progs which wipe out any of DL4-10 are allowed also to use d parameter positions for other purposes also. 120 to 125, however, are always preserved except when reading data, as described on next sheet.

120	EMPTY		
121	m		
122	n		
123	k		
124	BPA	}	A Matrix
125	<del>21</del>		
126	BP		
127	<del>Z</del>	}	d Diagonal
128	N		
129	R		
1210	BPR	}	R Scalar
1211	<del>ZR</del>		

All matrices and diagonals are preceded by a card

Y - m  
 X - n or N  
 O - B.P. or D.P.  
 1 - k (= row sum shift,) (= 1's in punched  
                                   binary                                  field, decimal)

both reading and punching. B.P. is always read into its respective store and k is always used on that occasion only. Note that the k in store refers to punching out only, and is read in, if required, by a special programme. All matrices and diags are, in fact, punched with  $2^{-k}$  row sums. Concerning m and n, for a matrix these are read in directly, but for a diagonal, m is ignored, and N only is read in. Scalars are punched on the Y row, with the B.P. on the O row.

B.P. are always stored and punched x P17, and D.P. are always stored and punched x P1. Thus it is easily noticed if by some error the use of a conversion factor after reading decimal data is omitted. The same stores in DL12 are used for B.P. and D.P. so that conversion is very easy.

Matrix Mult., and Diag. Pre - and Post-Mult progs examine the various m's and n's etc., and give a failure if these are not compatible.

### 3. Scheme B

#### Storage conventions:

All matrices and diagonals are stored in the same way on the drum and may start on any track. The elements are stored consecutively from m.c.4 onwards without any row sums, m.c.'s 0 to 3 being used to describe the matrix;

m.c.0 -  $\sum \sum$  (Grand sum of matrix, including m: n; B.P.)  
 m.c.1 - m x P17 (m = number of rows of matrix)  
 m.c.2 - n x P17 (n = " " cols. " " )  
 m.c.3 - B.P. x P17 (B.P. = binary places.)  
                                   (matrix in standard block-floating form.)

Diagonals are stored as 1 x n matrices.

Scalars are stored, as in Scheme A, in DL 129, 10, 11 as follows:

129 Scalar  
 1210 B.P. (of scalar)  
 1211  $\sum$  (sum of scalar and B.P.)

### 2. Read and Punch conventions

The same punching conventions of matrices and diagonals are used as for Scheme A (see Section I, 2) but reading and punching of intermediate results is of course not always necessary with this scheme.

Scalars are punched in the following manner:

Y - scalar (in binary)  
 X -  $\sum$  (Sum of words on Y & O rows.)  
 O - B.P. x P17 (binary places of scalar).

## Section II

## GENERAL INTERPRETATIVE PROGRAM (0107)

(INTERIM VERSION)

Date 25th August, 1955

Produced by M. Woodger  
B. W. MundayDescription

1. Up to 32 programs (comprising up to 64 trackfuls in all) are read onto the drum and called down into use by obeying successive interpretative codewords which refer to them by their number 1, 2, 3 ... in the order read in.

Each program can use all the high speed store except DL11, since the codes and interpretative program itself are on the drum while it is in use, and vice versa.

Up to 128 codes can be held in DL's 9 to 12, numbered 0, 1, 2 ...

2. Each program is provided just as normally read in (though column 54 must not be used), based on either  $l_0$  or  $8_0$  empty, filling any or all DL's except DL11, and entering a NIS DL in the usual way. It must meet three conditions.

- (i) It must preserve 1229, 30, 31 and end with 12-1 (32 mc), going to 130.
- (ii) It must clear TS13 at exit, except when the  $r^{\text{th}}$  program is invariably to be followed by the  $(r+1)^{\text{th}}$  or the  $s^{\text{th}}$  in which case TS13 is left containing P32 or S x P17 respectively.
- (iii) It must not write onto the last 16 tracks of the drum, or onto as many of the tracks adjacent to these as there are DL's of program stored, unless such programs are not required further. (The programs are stored backwards from track 14/15).

3. Each codeword is composed of four 8-digit numbers a, b, c, r. When  $r=0$  it means: call down and obey program no. c and proceed to the next code (if  $a=0$  it is ignored). When  $1 \leq r \leq 32$  it means: call down and obey program no. r, provide it with parameters a, b, c (each x P5) in QS 18<sub>0</sub>, 1, 2 respectively, and r x P17 in 18<sub>3</sub>, and proceed to the next code. When  $33 \leq r \leq 39$  the numbers a, b, c refer to the codewords themselves as follows:

r=33: Obey code no. c and continue from there.

r=34: { " " " c if code a is  $< 0$  } " " " "  
           { " " " b if " " "  $\geq 0$  }

r=35: { " " " c " " " "  $\neq 0$  } " " " "  
           { " " " b " " " "  $= 0$  }

r=36: Replace code no. c by code a + code b and proceed to the next code

r=37: " " " " " - " " " " " " "

r=38: " " " " code a " " " " " "

r=39: Read 32 codes to fill DLc then obey code b and continue from there.

(These codes occupy successive rows of three cards with the first four rows blank).

4. When r contains P31 the codeword is obeyed as a normal DEUCE instruction in m.c.30, provided that the NIS is zero and the timing no. is either 0 (causing the next code to be taken as usual) or 31 (causing a loop stop unless a successful discrimination occurs).

Such an instruction has access to the codes in DL's 9-12, and between any two such instructions (a) at most 9 ms. elapse, so that data may be read in from cards, and (b) all temporary stores are preserved. It can supply parameters to the next program to be called down by placing them

in	14	for TS14	or 44,5	for DS19
	15	" TS15	46,7	" DS20
	34	" TS16	48,9	" DS21
	30-3	" QS17		

If the next code has  $r=0$  the above instruction can also supply parameters directly in QS18.

5. Excepting codes with P31, a digit P32 either in a code or on the ID switches causes the machine to stop and display the code on the OPS lights before obeying it. It proceeds if given a single shot unless TIL key is depressed, in which case a codeword set on the ID switches is obeyed instead.

6. As each program is called down into the high speed store any DL, TS, DS or QS (other than DL1, DL11, TS13; also QS18 when  $r=0$ ) is left with whatever data it contained at the end of the previously called program. This feature is valuable for large programs involving repeated filling of the high speed store by successive "sections".

7. The general interpretative program automatically reads in codes 0.31 to fill DL9 just after reading the programs, and obeys first code 0. Further codes if needed are read in using a code with  $r=39$ .

#### 8. "Restore Control" Program (0109).

This program is usually stored as No.1 (i.e. in position 14/15 of the drum). If, at any time the programmer wishes to interrupt the prescribed codeword order - he may encounter a failure and wish to go back to the last codeword obeyed or may wish to insert a new code - he can do so by calling down the RESTORE CONTROL program in the following way:

Stop machine  
External Tree  
30-13 TT  
14-31 TT  
15-30 TT  
11-1 TT

NIS.1, TCI on until 1-1 appears on the instruction lights:

Internal (External tree off.)

If P32 is set on ID before calling down this program, (See Section III 6(11)) then putting the machine to "NORMAL" you can:

- (1) Observe the codeword displayed on OPS to find where the trouble occurred, or
- (2) go to any chosen codeword  $n$  by setting "0,0  $n$ , 33" on the ID and using the TIL key facility. (Section II, 5.)

#### Organisation of Program

Cards 1-3 clear the drum. Cards 4-12 constitute the program "Read Programs" which reads a data card P17 on Y row (card 13) followed by a copy of itself (cards 14-22), followed by the Interpretative Program proper (cards 23-46). The latter calls down "Read Programs" which proceeds to read the data card N, the N sections of program onto the drum from track 14/15 backwards, keeping a list of the intended DL's as destination numbers from the X rows ("Track Codes"), and a list of the entry words from the 1 rows of the last DL of each program "Program Codes". With the latter list is also incorporated the track number of the last track filled and the number of tracks filled for each program. A triple of cards with codes 0-31 is then read and code 0 obeyed.

The interpretative program contains special instructions to deal with the special codes with  $r \geq 33$ , and uses a Control Subroutine when a particular program is to be selected from the drum.

The control subroutine fills the "representative tracks" 15/1-15/12 with the corresponding contents intended for the selected program in DL's 1-12 respectively, excepting DL11, taking these from the program store in 14/15, 14, 13..., and using the Track Codes in 15/13, 14 and Program Codes in 15/15. It then contrives to interchange the contents of the high speed store (except DL11) and the representative tracks, and enter the selected program. The link orders are placed in 1229-31 by the Control Subroutine, and have the effect of calling down its DL1 from 15/1 and entering 10. The interchange of high speed store with representative tracks is then repeated, and the codes, Interpretative Program and Control Subroutine are again available for the next operation.

---

#### OPERATOR'S INSTRUCTIONS

Order of Cards. Program Cards 0 to 46 (Cards 1 to 3 clear drum and may be omitted if desired)

Data Card  
Controlled Programs  
Codes 0-31  
Data

Data Card Y row N x P17, specifies N sections of programs to be controlled.

Controlled Programs Each section of controlled program comprises triples of cards filling any DL's (other than DL11) in the usual way, assuming either 10 or 80 empty, and with the usual entry word on the 1 row of the first card of the last triple. Col. 54 must not be punched. Cards with zero in destination position on X row (e.g. initial or blank cards) are ignored between DL's.

Codes are supplied punched on successive rows of three cards with the first four rows blank.

Failure Indications None special to the Interpretative Program.

Stop Facility P32 set on ID switches, or P32 in any code, causes the machine to stop at 6, 1-lx having displayed the code on OPS without yet obeying it. Setting another code on ID switches, pressing TIL key, single shot, and TIL key off, then causes the new code to be obeyed instead. Otherwise a single shot causes normal continuation with the original code.





Section III An example of the use of Scheme B programs with General Interpretative Program for operations with matrices

1. The problem considered is that of improving the accuracy of a given approximate inverse  $B_0$  of a matrix A by use of the iteration formula

$$B_{n+1} = B_n(2I - AB_n)$$

where I is the unit matrix.

2. The following Scheme B programs suffice for this purpose:

1	Read Binary Matrix:	program occupying 3	tracks when stored on drum					
2	Punch Binary Matrix	"	"	3	"	"	"	"
3	Transpose	"	"	3	"	"	"	"
4,5	Subtract Matrices	"	"	8	"	"	"	"
6,7,8	Multiply Matrices, type AB'	"	"	16	"	"	"	"
				<u>Total</u>	<u>33</u>			tracks

These programs are enumerated according to the separate sections which they comprise thus "Subtract Matrices" consists of a small section (No.4) preceding "Add Matrices" (No.5), and sections 6,7,8 are the parts of "Matrix Multiplication" which form column sums, form double-length results, and round off results to single-length, respectively.

These programs use the parameters a, b, c of a codeword as follows:

"Read Matrix" stores a matrix beginning at track c (or simply "at track 0") as we shall say).

"Punch Matrix" punches out the matrix stored at track a, with row sums divided by 2<sup>0</sup>

"Transpose" stores at track c the transpose of the matrix at track a. Here a must differ from c and overlapping is not allowed in general, though the last track of the transpose can be track a.

"Subtract" stores at track c the difference A-B of the matrices at tracks a and b. There are no restrictions on a, b, c. In particular a matrix can be made zero by subtracting it from itself, and a second matrix copied into its place by subtracting the zero matrix; this is used here in the absence of a program to copy matrices.

"Multiply, AB'" first section, requires space from track 0 onwards for a row of double-length column sums of the matrix A at track a, and following the last track of these column sums an equal number of tracks for column sums of B (from track b). The second section of the program requires space from track 0 onwards for the double-length product matrix AB' which is written over the column sums. With the proviso that a, b, c are all larger than the last track used by the above double-length numbers there are no restrictions on a, b, c. c = 0 is also allowed but restricts the further use of the product matrix.

3. The calculation is expressed in the following steps:

First matrices 2I, A, B<sub>0</sub> are read into three places p, q, r on the drum, these being track numbers to be determined later so that the largest possible size of matrices can be used. The B<sub>0</sub>' the transpose of B<sub>0</sub>, is formed in a fourth place s,

B<sub>0</sub>' is multiplied by A to give B<sub>0</sub>'A' also at s,

B<sub>0</sub>'A' is subtracted from 2I to give 2I - B<sub>0</sub>'A' = (2I - AB<sub>0</sub>)' also at s,

$B_0$  is multiplied by the latter to give  $B_1 = B_0 (2I - AB_0)$  also at  $s$ ,

$B_1$  is copied into position  $r$  in place of  $B_0$  by the use of "Subtract" as described above, and the cycle is repeated.

It is arranged to punch out the matrices  $B_{n+1}$  and  $B_{n+1} - B_n$  every  $N+1$  cycles, where  $N$  is a constant provided as the last codeword.

4. In order to fix the track numbers  $p, q, r, s$  we allow the last 16 tracks of the drum for use by the Interpretative Program, the 33 tracks preceding those for the controlled programs, and sufficient tracks at the beginning of the drum to accommodate one double-length product matrix (the column sums, occupying considerably less space than a product matrix, can be neglected). We have thus 206 tracks for approximately six square matrices, or 34 per matrix. A square matrix of order 33 occupies  $33^2 + 4$  words, i.e. 35 tracks, so the limit is order 32. Each matrix of order 32 uses 33 tracks and a double-length one just 64 tracks. We therefore choose  $p = 64, q = 97, r = 130, s = 163$ , and have tracks 196 to 205 free for further programs if wanted.

5. The detailed codes are as below, and include illustrations of the use of the special operations  $r = 33, 35, 37, 38$ . (See GIP (iii)). Only one DL is used for codes.

DL9 m.c. No.	a	b	c	r	
0	0	0	64	1	Read in 2I
1	0	0	97	1	Read in A
2	0	0	130	1	Read in $B_0$
3	21	0	19	38	Set counter $x = N$
4	130	0	163	3	Form $B_0^t$
5	163	97	163	6	Form $B_0^t A^t$
6	64	163	163	4	Form $2I - B_0^t A^t$
7	130	163	163	6	Form $B_1$
8	19	13	9	35	Examine counter. If zero go to code 13
9	130	130	130	4	Form zero matrix
10	163	130	130	4	Set $B_1$ in place of $B_0$
11	19	20	19	37	Reduce counter $x$ by one
12	0	0	4	33	Go back to code 4
13	163	130	130	4	Form $B_1 - B_0$
14	163	0	0	2	Punch out $B_1$
15	130	0	0	2	Punch out $B_1 - B_0$
16	130	130	130	4	Form zero matrix
17	163	130	130	4	Set $B_1$ in place of $B_0$
18	0	0	3	33	Go back to code 3
19	0	0	0	0	(Counter $x$ )
20	0	0	1	0	(Unit for counting)
21	0	0	N	0	(Parameter for controlling punching frequency)

## 6. Notes

(i) In setting up a problem for the first time it is useful to keep P32 on the ID switches, to stop and check each codeword before it is obeyed.

(ii) It is worth the extra track to incorporate "Restore Control" as program No.1 in case overwriting of matrices causes the machine to fail in a controlled

program on the first trial run. One can then set P32 on ID, restore control, examine the displayed codeword on OPS to find at what point the trouble occurred, and go to any chosen codeword  $n$  by setting "0, 0,  $n$ , 33" on ID and using the TIL key facility (see GIP (5) Section II).

(iii) When testing codeword sequences it is useful to begin them at m.c. 8 (top of second card of codes) and have manual control of successive codes from ID switches available at m.c. 0 by use of the following codes:

m.c.	0	0, 0-15, 0, 0X, with P31	normal DEUCE instructions (see GIP (4))
	1	0, 0-92, 2, 0, with P31	
	2	blank	
	3	0 0 0 33	

Here the first code is a dummy to give a recognisable word "0-15" on control-lights. "0-9<sub>2</sub>" reads a codeword set on ID to code no.2 and obeys it as soon as a oneshot is given, but "0-9" on control-lights would have been confusable with the effect of reading in 32 new codes to DL9 (See GIP (3), Section II, for  $r = 39$ ).

Using this method the machine stops at "0-15" and manual control from ID is available. When required, the main codeword sequence at m.c.8 is entered by setting 0, 0, 8, 33 on ID and giving oneshot.

(iv) In the above example of matrix operations it should be noted that the process will converge only if  $B_0$  is such that  $(I - AB_0)^n$  tends to the zero matrix as  $n \rightarrow \infty$ . In many practical cases this is true even when the unit matrix itself is chosen for  $B_0$ .

(v) The complete checking and automatic control of binary places in these Scheme B matrix programs results in a sacrifice of speed. One cycle of the above process for order 7 took almost 40 seconds, plus about 8 seconds for punching out. Reading in programs and matrices took 1 minute.



Section IV      Identification Numbers of Scheme A and B programs

28 December, 1955

General

- 0107      General Interpretive Program (Second Version dated 29.11.55, Interim Version 23.8.55).
- 0109      Restore Control.
- 0112      Auxiliary Code Program.
- 0113      Replace Programs, incorporating Restore Control.

Scheme A

- 1101      Read Binary Matrix.
- 1102      Read Diagonal.
- 1103      Read Scalar (Latest version dated 14.12.55).
- 1105      Read Decimal Integer Matrix, punching error identification.
- 1200      Matrix Add.
- 1203      Matrix Mult.  $AB'$  (Punches Double-length results),  $n \leq 96$ .
- 1202      Residuals (L.R.)
- 1205      Diag. Postmult.
- 1206      Diag. Premult.
- 1207      Form Column Sums,  $n \leq 128$ .
- 1208      Shift Matrix (Rounds off Product Matrix to single length).
- 1210      Prepare for Shift matrix 1211.      Restart Shift matrix.
- 1218      Scalar Mult.
- 1219      Transpose,  $mn \leq 4096$  (Not usable with Control).
- 1222      Form Conversion Factor (after reading decimal data).
- 1212      Reverse Signs of Matrix.
- 1224      Matrix Mult.  $AB'$ ,  $n \leq 128$ .
- 1225      Check Matrix Store.
- 1227      Square Scalar.
- 1300      Punch Binary Matrix.
- 1301      Punch Diagonal.
- 1302      Punch Scalar (Latest version 14.12.55).
- 1305      Punch Grand Sum.

1306 Punch Transpose of Matrix.

1500 Latent roots ( $n \leq 32$ )

Scheme B

2100 Read Binary Matrix.

2102 Read  $\frac{1}{2}mn$  Scalars as Floating Matrix.

2200 Matrix Mult.  $AB'$ .

2201 Diag. Postmult.

2202 Scalar Mult. (Latest version 19.12.55; two Codeword version 19.10.55).

2203 Transpose.

2204 Matrix Subtract.

2205 Matrix Add.

2206 Punch Nonzero Tracks a to c-1.

2207 Comp. Cols.

2208 Fetch Scalar.

2209 Diag. Premult.

2210 Comp. Rows.

2215 Extract Submatrix.

2216 Extract Diagonal.

2217 Select Element.

2218 Insert Zero Rows and Cols.

2219 Expand Diagonal.

2220 Shift up or down.

2221 Blockfloat.

2222 Reciprocal (of  $1 \times n$  matrix, element by element).

2223 Square Root (of  $1 \times n$  matrix, element by element).

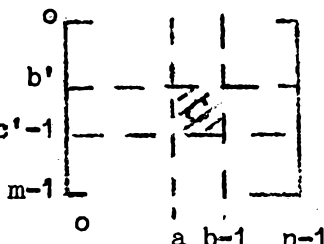
2300 Punch Binary Matrix.

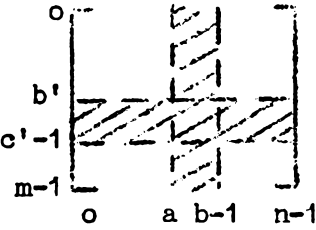
2302 Punch Matrix for Latent Roots (shifted down 2 places, with  $1/64$  row sums and grand sum).

No.	Program	Tracks Used	Codeword In general(a b c r)	Restrictions	Other Information (Data, etc.)
2100	Read Binary Matrix (1 Section)	3	$\begin{matrix} & o & o & \underline{c} & r \\ \text{Read Matrix in binary} & & & & \\ \text{to track } \underline{c} & & & & \\ \text{Elements are punched 12} & & & & \\ \text{per card with exact row} & & & & \\ \text{sums. Each row starts} & & & & \\ \text{on a new card.} & & & & \end{matrix}$		<p>1. <u>Data Card.</u>  <math display="block">\left. \begin{matrix} Y \dots m \\ X \dots n \\ 0 \dots \text{b.p.} \\ 1 \dots k \end{matrix} \right\} \times P17</math> <math>k = \text{row sum shift}</math>  This card precedes matrix</p> <p>2. <u>Failures</u>  <math>\begin{matrix} 3 &amp; 21-29 &amp; \text{Error on} \\ &amp; &amp; \text{OFS lights} \\ &amp; &amp; \text{Row sum wrong} \\ &amp; &amp; \text{Actually one of cycle} \\ &amp; &amp; \text{of three,} \\ 3 &amp; 21-29 &amp; \\ 1 &amp; 21-28 &amp; \\ 1 &amp; 9-24 &amp; \end{matrix}</math></p>
2102	Read $\frac{1}{2}$ mm Scalars as Fully Floating Matrix (1 section)	3	$\begin{matrix} & o & o & \underline{c} & r \\ \text{Reads } \frac{1}{2}\text{mm scalars} & & & & \\ \text{and stores them as an} & & & & \\ \text{(m x n) fully floating} & & & & \\ \text{matrix at track } \underline{c}. & & & & \end{matrix}$	N.B.1. Scalars are punched one per card in the following way (Y . scalar in binary (X . sum of Y80 rows (O . b.p. x P17	<p>1. <u>Data Card</u>  <math display="block">\left. \begin{matrix} Y \dots m \\ X \dots n \end{matrix} \right\} \times P17</math> card precedes the matrix</p> <p>2. Let <math>x_{ij}</math> be typical scalar; if <math>x_{ij} = a_{ij} \times 2^{bij}</math>  <math display="block">\begin{matrix} (-2^{32} &lt; b_{ij} \leq 2^{32}) \\ \frac{1}{2} &lt; a_{ij} &lt; 1 \text{ if } a_{ij} &gt; 0 \\ -1 &lt; a_{ij} &lt; -\frac{1}{2} \text{ if } a_{ij} &lt; 0 \end{matrix}</math> then in resultant matrix <math>b_{ij}</math> and <math>a_{ij}</math> are stored side by side.</p>
2200	Matrix Multiplication (forms $AB'$ ) (3 sections)	16	$\begin{matrix} & a & b & o & r \\ \text{Mults. matrix A, stored} & & & & \\ \text{at } \underline{a} \text{ by } B' \text{ the trans-} & & & & \\ \text{pose of matrix B, at } \underline{b}; & & & & \\ \text{result } AB' \text{ is stored at} & & & & \\ \underline{c}. & & & & \end{matrix}$	N.B. $\left. \begin{matrix} (1) a = b \\ (2) c = a \text{ and } c = b \\ (3) c = o \text{ allowed but} \end{matrix} \right\}$ are allowed and restricts further use of the product matrix.	All stages are fully checked. Failure may occur with certain matrices if the number of columns exceeds 256 but is unlikely for normal matrices. a, b, c must normally be larger than (1) the number of tracks required to store double-length product matrix (beginning at 0/0) (2) the sum of the number of tracks required to store rows of double-length column sums of matrices A and B Failures: details of these appear in Deuce programme 73.
2201	Diagonal Postmult. (2 sections)	9	$\begin{matrix} & a & b & c & r \\ \text{Mult. matrix A at } \underline{a} & & & & \\ \text{by diag. matrix B} & & & & \\ \text{(stored as } 1 \times n & & & & \\ \text{vector) at } \underline{b}; \text{ result} & & & & \\ \text{C = AB is stored at } \underline{c}. & & & & \end{matrix}$	1. $c \neq a$	1. "Diagonal Postmult." needs space for double-length col. sums of A at position $\underline{c}$ . Hence $a \neq c$ .

No.	Program	Tracks Used	Codeword In General (a b c r)	Restrictions	Other Information (Data, etc.)
2202	1. Scalar Mult. (1 Section) New version; dated 19.12.55.	7	$\begin{matrix} a & o & c & r \\ \text{Multiplies matrix A at } a & & & \\ \text{by scalar in Scheme A} & & & \\ \text{scalar store (D.L.125,10,11);} & & & \\ \text{result is stored at } c. & & & \end{matrix}$	N.B.1. Usually used in conjunction with "Select Element" (2217) or "Fetch scalar" (2208)	1. Both programs (2217 and 2208) read down to the Scheme A store (ready for this Scalar Mult. program), a scalar selected from a matrix stored on the drum.
2202	2. Scalar Multipli- cation (old version) (Dated 19.10.55)	7	$\begin{matrix} a' & b' & c' & r' \\ a & b & c & r \end{matrix}$ Scalar number <u>b</u> (bth) of matrix A at <u>a</u> , multiplies matrix at <u>c</u> ; result is stored at <u>c'</u>	1. r' is serial nu. of the "AUX. CODE" program r is the serial nu. of the scalar mult. program. 2. The elements of matrix A are numbered from b=0	N.B.1. The "AUX. CODE" program is usually stored as the 2nd controlled program if "Restore Control" is the first.
2203	Transpose Matrix	3	$\begin{matrix} a & b & c & r \\ \text{Stores at track } c & & & \\ \text{the transpose (A')} & & & \\ \text{of a} & & & \\ \text{matrix (A) at track } a. & & & \end{matrix}$	(1) $c \neq a$ . However the last track of the transpose can be track a.	
2204	Matrix Subtract (2 sections) [Must be followed by Matrix ADD (2205), which provides the 2nd section.]	1 +7	$\begin{matrix} a & b & c & r \\ \text{Stores at } c & & & \\ \text{the difference (A-B) of} & & & \\ \text{matrix A at } a & & & \\ \text{and} & & & \\ \text{matrix B at } b. & & & \end{matrix}$	No restrictions on a, b, c.	1. In particular, a matrix can be made zero by subtracting it from itself, and a second matrix copied into its place by subtracting the zero matrix; mcs, 0-3 will be preserved.
2205	Matrix Add (1 section)	7	$\begin{matrix} a & b & c & r \\ \text{Adds matrices A (at} & & & \\ \text{a) and B (at } b); & & & \\ \text{result is stored} & & & \\ \text{at } c. & & & \end{matrix}$	No restrictions on a, b, c.	
2206	Punch Non-Zero Tracks	1	$\begin{matrix} a & o & c & r \\ \text{Punches out all non-} & & & \\ \text{zero tracks } a, a + 1, & & & \\ \dots c-1 \text{ omitting any} & & & \\ \text{earlier ones.} & & & \end{matrix}$	(1) Gives track no.XP17 on the top row of each triad of cards.	
2207	Compound columns. (forms [A:B])	7	$\begin{matrix} a & b & c & r \\ \text{Takes matrices A at } a & & & \\ \text{\& B at } b \text{ and stores} & & & \\ \text{[A:B] consecutively} & & & \\ \text{at } c. & & & \end{matrix}$	(1) a=c not allowed, unless result happens to occupy 1 track (in general)	1. The rows, originally of matrix B succeed the corresponding rows of matrix A in the compounded matrix stored at <u>c</u> .



No.	Program	Tracks Used	Codeword In General (a b c r)	Restrictions	Other Information (Data, etc.)
2208	Fetch Scalar	1	a b o r Reads down to Scheme A scalar store (D.L. 129, 1C, 11) the $b$ 'th element of matrix at a (first element $b=0$ )	1. This program uses 1 track less than 2217 and behaves like 2217 with $c \neq 0$	1. The "fetched" scalar is ready for use in the "Scalar Mult" program. See 2202 (1st. version) 19.12.55.
2209	Diagonal Pre-Mult	7	a b c r Matrix A (diag.) at a and multiplies matrix B at b; result, C is stored at <u>c</u> .	1. $c=a$ allowed 2. c may displace B	1. Failures: 4 24-31 X Shift too big 3 24-31 X Distributive check failures. 2. Matrix sum wrong (A 1 29-31 X (B 6 29-31 X
2210	Compound Rows $\begin{bmatrix} A \\ B \end{bmatrix}$	6	a b c r Takes matrix A at a and matrix B at b; compounded matrix stored consecutive-2. c must not overlap a or b.	1. $a \neq c$ (unless result happens to occupy 1 track. 2. c must not overlap a or b.	1. First row of matrix B succeeds last row of matrix A without a break in storage.
2215	Extract Sub-Matrix	4	$\begin{matrix} a'b'c'r' \\ a b c r \end{matrix}$ Extracts from matrix A ( $a_{ij}: 0 \leq i, m-1$ $0 \leq j, n-1$ ) the submatrix, C ( $a_{ij}: b' \leq i, c'-1$ $a \leq j, b-1$ ) B stored at <u>c</u> .	1. Requires AUX. CODE PROGRAM $r'$ is serial nu. of AUX. CODE PROGRAM (0112) and $r$ is serial nu. of program 2215. 2. (N.B. $0 \leq b' \leq c'$ $0 \leq a \leq b$ ) $c=a'$ allowed	1. Suppose A is $(m \times n)$ matrix; rows numbered 0 to $m-1$ and cols. 0 to $n-1$ .  The matrix C defined as shown may then be extracted using this program (with 2 codewords).
2216	Extract Diagonal	4	a o c r Extracts principal diagonal of matrix A at a and stores it at c as $(1 \times n)$ matrix (row vector).	1. $c=a$ allowed. 2. A need not be square.	

No.	Program	Tracks Used	Codeword In General (a b c r)	Restrictions	Other Information (Data, etc.)
2217	Select Element	2	a b c r Reads down to Scheme A store (D.L. 129, 10, 11) the b'th element of matrix A at <u>a</u> . (first element given by b=0)	1. $c=0$ } treats matrix $c \neq 0$ } in standard block-fully floating form. 2. Operation of "Fetch Scalar" is equivalent to this with $c \neq 0$ .	1. Matrix A may therefore be either in standard (i.e. block-floating) form or fully floating form. (Consecutive pairs of elements being regarded as the two parts of standard floating binary numbers.)
2218	Insert Zero Rows and Columns	4	a' b' c' r' a b c r Insert in matrix A, at <u>a'</u> (elements $a_{ij}$ ). $0 \leq i \leq m-1$ $0 \leq j \leq n-1$ Zeros in rows and cols. $b' \leq i \leq c'-1$ $a \leq j \leq b-1$	1. Requires AUX.CODE PROGRAM; r' is serial nu. of AUX.CODE PROGRAM (0112) (usually $r' = 2$ ) and <u>r</u> is the serial number of program 2215 2. $c' \neq a$	1. Let A be an $m \times n$ matrix;  rows are numbered from 0 to $m-1$ and cols. from 0 to $n-1$ . The shaded array of zeros may then be inserted using this program in the manner described.
2219	Expand Diagonal	4	a o c r (1 x n) matrix at <u>a</u> is stored as the diagonal of an (n x n) matrix, at <u>c</u> .	1. $c \neq a$	1. Effect is to border the diagonal with zeros.
2220	Shift up or down	4	a b c r Matrix at <u>a</u> shifted down <u>b</u> places and rounded off; result is stored at <u>c</u> .	1. $c = a$ allowed	1. (Shifted down if $1 \leq b \leq 32$ . (Shifted up (-b places) if $-31 \leq b \leq -1$ . For $b > 32$ } elements of the matrix become zero. $b < -31$ ) For $b = 0$ , the matrix is unaltered. (This gives a method of copying a matrix).
2221	Block Float	4	a b c r Converts matrix at <u>a</u> (in fully floating form) to standard block - floating form; result is stored at <u>c</u> .	1. $c = a$ allowed.	

No.	Program	Tracks Used	Codeword In General (a b c r)	Restrictions	Other Information (Data, etc.)
2222	Reciprocal (of 1 x n matrix) (- element by element)	5	<p style="text-align: center;">a o c r</p> Evaluates the reciprocal of each element of the (1 x h) matrix stored at <u>a</u> ; result is stored at <u>c</u> .	1. c=a allowed	1. If the matrix at <u>a</u> has elements $a_i$ ( $1 \leq i \leq n$ ) then the elements $c_i$ , ( $1 \leq i \leq n$ ) of the "reciprocal" matrix are given by $c_i = \begin{cases} a_i^{-1} & \text{if } a_i \neq 0 \\ 0 & \text{if } a_i = 0 \end{cases}$
2223	Square Root (of 1xn matrix, element by element)	4	<p style="text-align: center;">a o c r</p> Evaluates the square root of each element of the (1xn) matrix stored at <u>a</u> ; result is stored at <u>c</u> .		1. If matrix at <u>a</u> has elements $a_i$ ( $1 \leq i \leq n$ ) then the elements $c_i$ ( $1 \leq i \leq n$ ), of the "square root" matrix, are given by $c_i = \sqrt{a_i} \quad (a_i \geq 0)$ Failure for $a_i < 0$ . Truncation error between 0 and P1.
2300	Punch Binary Matrix	3	<p style="text-align: center;">a o c r</p> Punches out the matrix stored at <u>a</u> with row sums divided by $2^c$ .		
2302	Punch Matrix for L.R.60. (Latent roots program)	3	<p style="text-align: center;">a o o r</p> Punches out the matrix at <u>a</u> , with elements shifted down 2 places and $2^{-6}$ (row sums) and grand sum (unscaled).		1. The output is then in the form required for the L.R.60 program.

