Technical Newsletter No. 6

APPLIED SCIENCE DEPARTMENT

IBM

APPLIED SCIENCE DIVISION

Technical Newsletter No. 6

November 1953

We wish to extend thanks to the authors of these papers for their cooperation in contributing to this interchange of technical information.

CONTENTS

THE IBM 602-A CALCULATING PUNCH AS A GENERAL PURPOSE CALCULATOR

Robert W. Schrage

Esso Standard Oil Company
East Coast Technical Service Division
Linden, New Jersey

INTRODUCTION

Some time ago a study was initiated with the object of adapting catalytic cracking yield and quality predictions to IBM calculators. It was soon recognized that the Card-Programmed Electronic Calculator (CPC) was best suited to this type of work. However, since one of these machines was not immediately available to the author, a system was developed for doing these calculations on the simpler 602-A calculator.

Catalytic cracking yield and quality predictions are typical of many technical calculations. They consist of two basic elements: first, charts or tables which express empirical functions of certain base variables must be consulted; second, values of these functions must be combined according to certain specified arithmetic rules to yield final answers. This paper describes a method by which both of these operations are done on the 602-A calculator in an integrated fashion.

Two general purpose control panels for the 602-A are involved in this work. The first panel permits card-programming of simple arithmetic calculations involving only addition, subtraction, multiplication, and division. Sequential calculations of any length can be performed provided no more than seven numbers must be stored at any step. The second control panel is for performing interpolations in third difference tables of functions of single variables. These tables serve as a highly condensed form of "correlation library" on punched cards.

The two calculation systems for the 602-A, that is card programming and table look-up, have been designed to work together. Interpolated values calculated from the "library" of difference tables serve as the intermediate data on which the final card-programmed calculations are based. No manual handling or key punching of these intermediate data is required.

Although the time saved by the machine calculations, as compared to previous manual methods, is in itself of considerable importance, the release of technical personnel for other work and the accuracy and consistency of the calculated results are probably of even greater importance. While this system of calculation for the 602-A has been primarily a temporary expedient, until use could be made of the faster and more flexible CPC, it will probably be interesting to small scale users of computing equipment, where the volume of work to be done is insufficient to justify the more expensive machine.

CARD PROGRAMMING SET-UP

The control panel to be discussed is an adaptation of one described by E. V. Hankam[1] of the Watson Scientific Computing Laboratory. Certain features of the Hankam board, for shifting of numbers, have been eliminated, resulting in some simplification of wiring with little loss in flexibility for many types of problems.

The control panel described here is based on the premise that the calculator will read all data in the form of eight-digit numbers, having the decimal point located between the fourth and fifth digits. Such numbers might be indicated as xxxx.xxxx in notational form. All answers which are punched by the 602-A are exactly similar eight-digit numbers.

Possible Operations

The operation of the calculator is under the control of code punches in the cards which it reads. A group of cards for performing a sequential calculation is called a program deck, and punches in these cards may cause any of the following operations to be performed:

(1) A number read from a card may be placed in any one of seven storage units in the machine.

(2) A number on a card may be added to, multiplied by, or divided into a number already in a storage unit. Furthermore, a number in storage may be subtracted from a number read from a card. The result of any of these operations may be placed in any of the machine's seven storage units.

(3) Any two numbers stored in the calculator may be added, subtracted, multiplied or divided and the answer can be read into any desired storage unit.

(4) At any point in a calculation sequence an answer can be punched into a card.

(5) All operations are done with full decimal and sign control provided conventions for feeding data to the machine are

1. Hankam, Eric V., A Card Programmed 602-A Board - 9 pp. mimeographed New York: IBM, November 25, 1951

followed. No provision is made for rounding, however, and care must be taken in the sequencing of calculations so that numbers do not exceed the eight-digit capacity of the machine by becoming so large or so small that significant figures are lost.

Card Layout and Coding

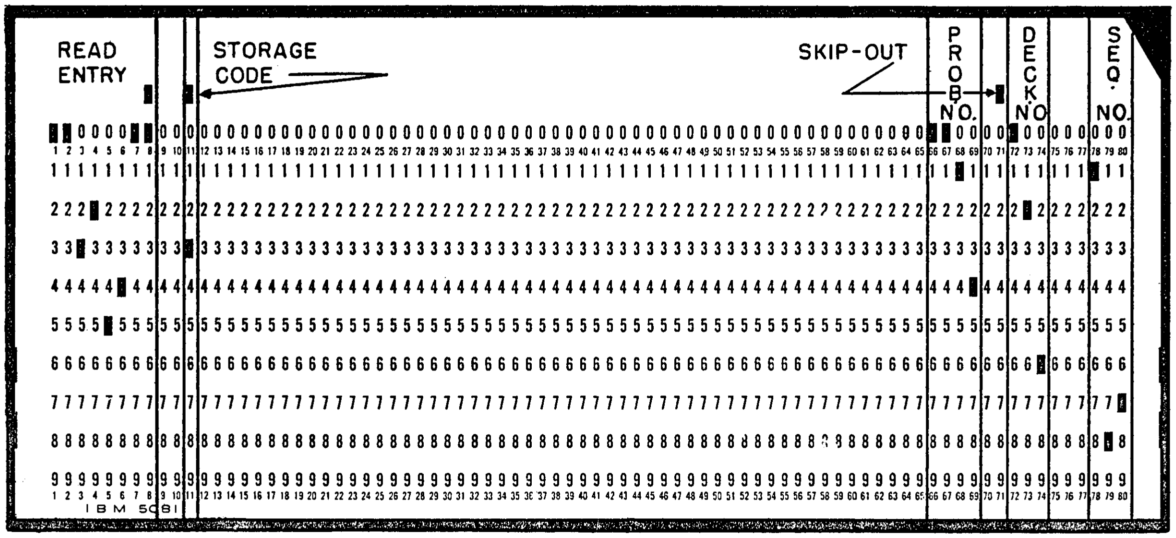


Fig. 1 Data Card

Fig. 1 shows a typical data card for performing the type of operation described in (1) above. The first eight columns of the card contain the number to be stored in the calculator. An X punch in the eighth column is used to denote negative numbers. The eleventh column carries an X punch which indicates that on this type of card the number in Cols. 1-8 is simply to be stored. A digit punch from 2 to 8 in Col. 11 designates which of the seven available storage units is to be used.[2] Col. 71 carries an X punch which causes the card to be "skipped out" of the machine without punching since it is not an answer card. Cols. 66-69 identify the Problem Number of the calculation with which this card is associated. Cols. 72-74 designate the card program deck from which the card comes, and Cols. 78-80 designate the Sequence Number of the card in that program deck.

Fig. 2 shows an instruction card involved in the type of operation discussed in (2) above. Cols. 1-8 again carry the number to be read in from the card. Col. 14 specifies according to a digit code whether this number is to be added to (1) multiplied by (3) or divided into (4) a second number held in a storage unit designated by a digit (called the B-code) in Col. 15.

2. Storage unit 1 is reserved for calculating functions and cannot be used for storage.

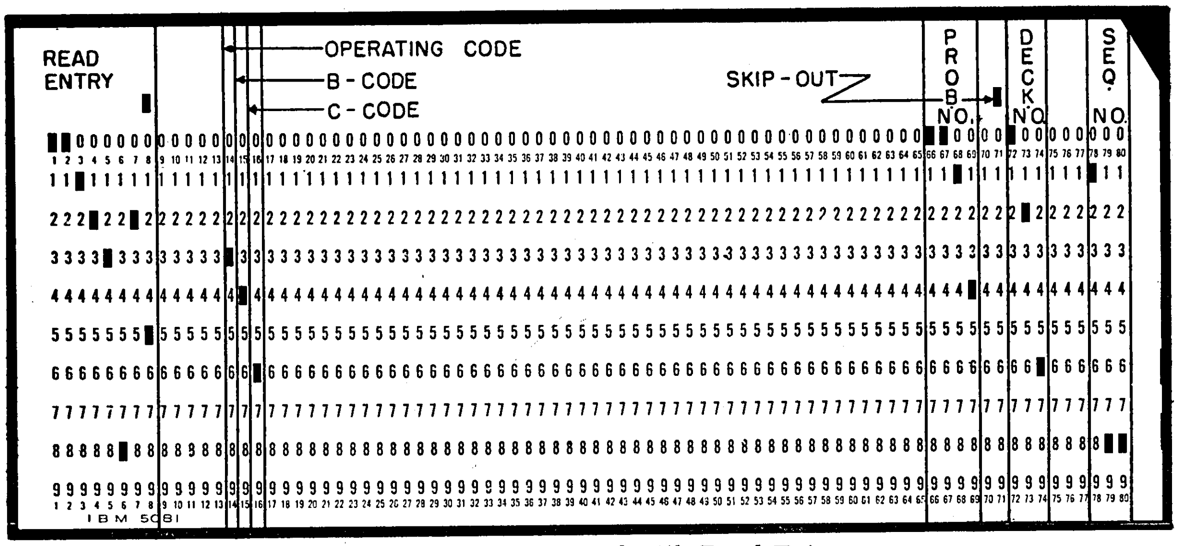


Fig. 2 Instruction Card with Read Entry

Col. 16 specifies (by a digit called the C-code) the storage unit into which the result of the operation is to be placed. If a number in a storage unit is to be subtracted from the number on the card, then Col. 14 should carry a 2 punch.

The card shown in Fig. 2 specifically calls for the number in Cols. 1-8 (-12.3825) to be multiplied (3 in Col. 14) by the number in the fourth storage unit (4 in Col. 15) and the result read into the sixth storage unit (6 in Col. 16). All of the punches in the last columns of the card have the same significance as in Fig. 1.

Fig. 3 shows the third type of card. This card contains no number in Cols. 1-8, since it will deal only with numbers already stored in the machine. The operation to be

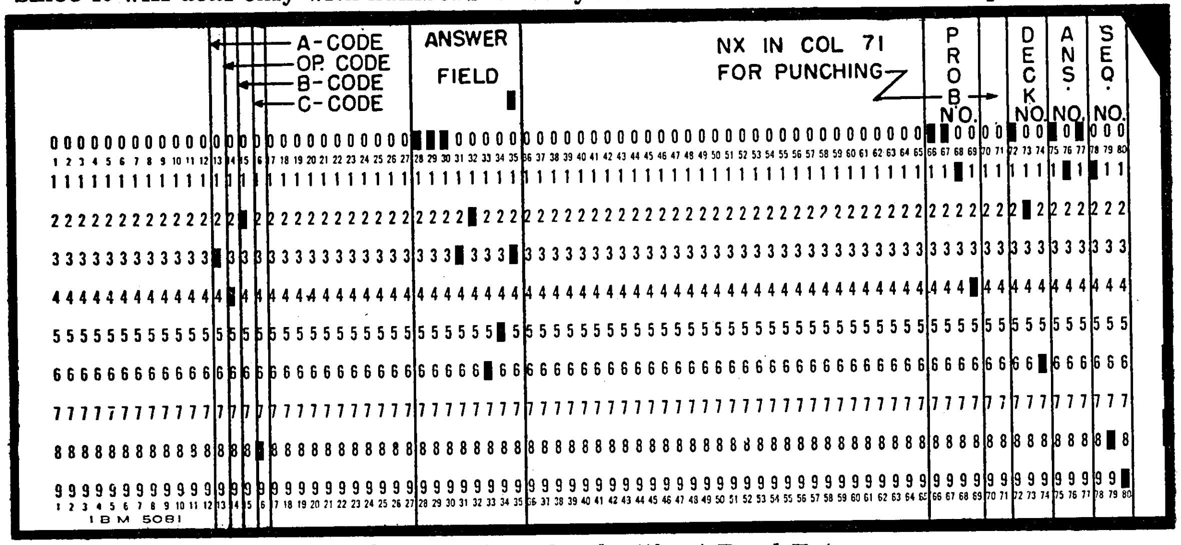


Fig. 3 Instruction Card without Read Entry

performed is of the type A ⊙ B = C.[3] Col. 13 carries the A-code (storage unit in which A is held), Col. 14 contains the operating code (1 for addition, 2 for subtraction, 3 for multiplication, and 4 for division), and Cols. 15 and 16 carry the B- and C-codes

3. The symbol ⊙ denotes any of the four possible basic arithmetic operations (+, -, ×, ÷).

respectively. In subtraction the first factor, A, must always be the minuend; in division A must always be the divisor.

On the card shown in Fig. 3 the answer is to be punched into the card. For this purpose the eighth storage unit must be used and the X punch must be omitted in Col. 71. The manner in which the control board is wired requires that any time a number is read into the eighth storage unit an answer must be punched. The number then remains in the storage unit for further calculations if needed.

Answers are punched into Cols. 28-35. If the answer is a negative number it will carry an X in Col. 35. An Answer Number is designated in Cols. 75-77. All other code designations in the latter part of the card are the same as in Fig. 1.

Detailed wiring directions for the 602-A card-programmed control panel are given at the end of the article and the programming of a simple illustrative example is shown.

Scope and Speed of Operations

The problems for which this control panel has been used thus far involve program decks of about two hundred cards. A program deck of this size averages about ten minutes running time on the 602-A, and any number of successive related or unrelated program decks can be fed continuously to the calculator.

INTERPOLATION

A frequently recurring problem in technical calculations by IBM machines is that of representing correlations in a form adaptable to machine treatment. If the equation of a correlation is known or can readily be obtained it is generally easiest to evaluate the equation directly through card-programming. If the equation is not known, however, or it is known but extremely cumbersome, then some thought must be given to alternative solutions.

There are at least two straight-forward choices open:

1. An approximate analytic expression for the correlation may be devised.
2. The correlation may be represented in tabular form.

Which of these choices is adopted must be decided largely upon the basis of circumstances governing a particular instance. Let it suffice to say at this point, however, that it has frequently been found that tabular representation seemed most convenient and satisfactory.

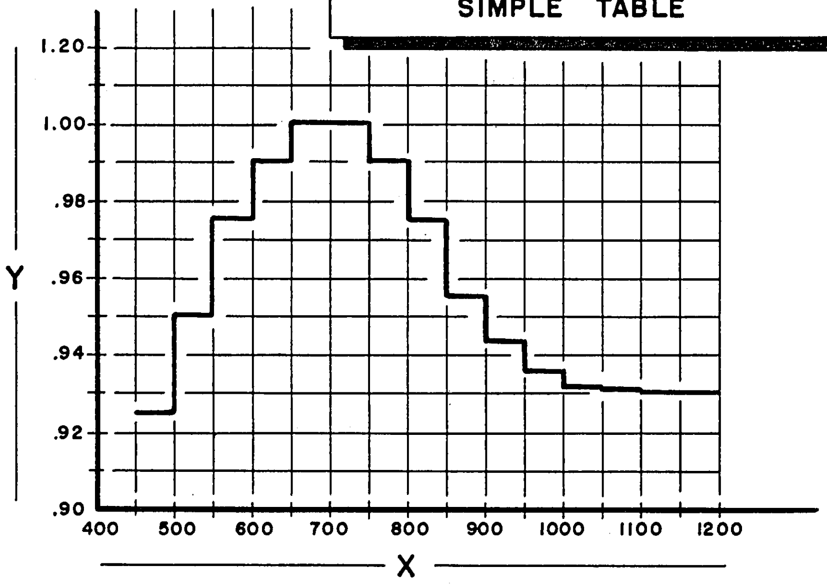
This report considers tables of functions of a single variable. Later work (probably with the Card-Programmed Electronic Calculator) may possibly extend these techniques to tables of functions of two or three variables. For the present, however, this latter type of machine problem has been handled through analytic expressions or not at all.

After having decided to express a function in tabular form there remains a problem as to exactly what kind of a table should be produced. For the sake of consistency (which has a very important significance in this work) a single type of table is much to be desired. It was decided that third difference tables represented the best compromise between several factors considered, namely: ease of preparation, speed of machine calculations, and acceptability of results. The following section discusses briefly what difference tables are and how they can be constructed.

Difference Tables

The simplest type of table which can be devised is one which contains a single value of the function under consideration for each value of the argument which is tabulated. A table of this type is shown in the first two columns of Table I. When a table such as this is used in the form of IBM cards, any intermediate value of the argument will select the value of the function corresponding to the next lowest (or, if desired, the next highest) value of the argument. If all the intermediate values which could be "calculated" from such a table were plotted in graphical form, the result might be represented by the stepwise curve shown in Figure 4. Even though the resulting curve may represent the actual function within tolerable limits of absolute accuracy, nevertheless the erratic manner in which the calculated values vary is quite objectional in studies involving the effects of small changes in the argument.

FIG. 4 GRAPHICAL REPRESENTATION OF SIMPLE TABLE



A better table to use would be a "first difference" table. As shown in the first three columns of Table I, this type of table lists the differences between successive

TABLE I. VARIOUS TYPES OF DIFFERENCE TABLES

← Third Difference Table →				
← Second Difference Table →				
← First Difference Table →				
← Simple Table →				
Argument, x	Function, y	1st Diff, $\Delta^1 y$	2d Diff, $\Delta^2 y$	3d Diff, $\Delta^3 y$
450.0000	.9250	.0250	.0000	-.0100
500.0000	.9500	.0250	-.0100	.0050
550.0000	.9750	.0150	-.0050	-.0050
600.0000	.9900	.0100	-.0100	.0000
650.0000	1.0000	.0000	-.0100	.0050
700.0000	1.0000	-.0100	-.0050	.0005
750.0000	.9900	-.0150	-.0045	.0121
800.0000	.9750	-.0195	.0076	-.0036
850.0000	.9555	-.0119	.0040	.0002
900.0000	.9436	-.0079	.0042	-.0020
950.0000	.9357	-.0037	.0022	-.0012
1000.0000	.9320	-.0015	.0010	-.0005
1050.0000	.9305	-.0005	.0005	-.0005
1100.0000	.9300	.0000	.0000	-
1150.0000	.9300	.0000	-	-
1200.0000	.9300	-	-	-

values of y as well as the values themselves. In using such a table, one first calculates

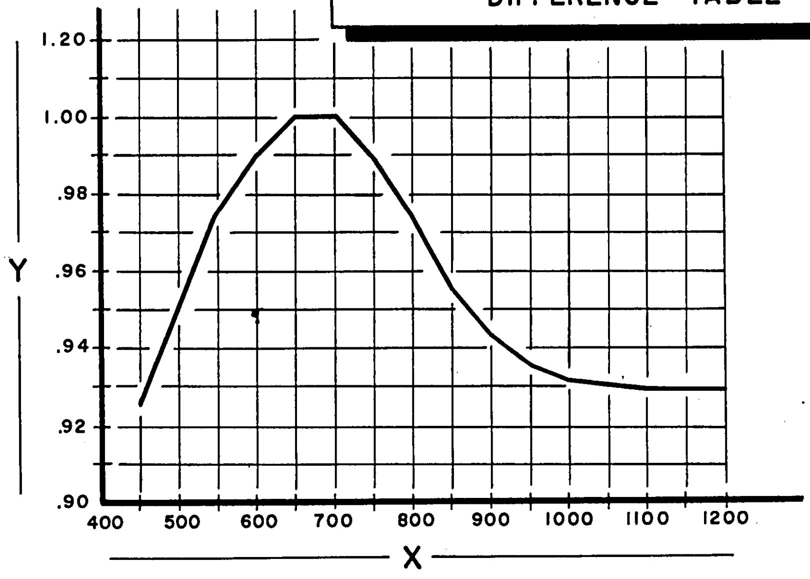
$$p = \frac{x - x_n}{\Delta x} \tag{1}$$

where x_n is the first value of the argument which precedes x in the table and Δx is the constant interval between the tabulated values of the argument. The value of y corresponding to x is then

$$y = y_n + p\ \Delta^1 y_n \tag{2}$$

This is equivalent to ordinary linear interpolation. A graphical representation of the first difference table in Table I is shown in Figure 5.

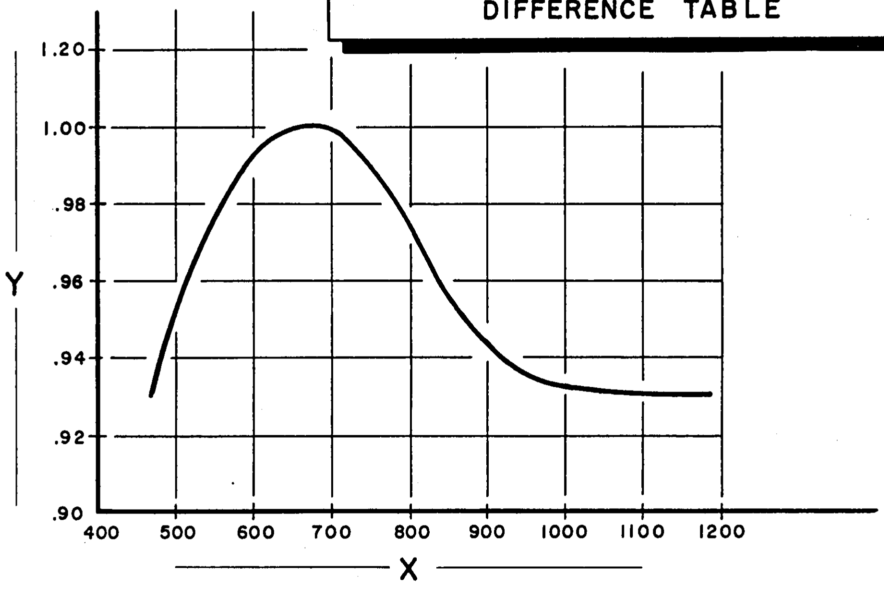
FIG. 5 GRAPHICAL REPRESENTATION OF FIRST DIFFERENCE TABLE



One may resort to higher order differences to obtain smoother representations of the required function. Second and third difference tables are also shown in Table I. The second difference is the difference between first differences, the third difference is the difference between second differences, and so on. Figure 6 shows a graphical representation of the third difference table given in Table I. In order to make use of higher order differences one must use increasingly involved interpolating procedures. This question will be examined in a moment, but first another aspect of the problem will be considered.

Obviously any type of table can be made a more accurate approximation of the function it represents by increasing the number of values tabulated (i.e. decreasing tabular interval). However, there is a real disadvantage associated with this, particularly when tables are to be expressed on IBM cards. The number of cards necessary goes up in proportion to the number of values tabulated and in many cases to obtain an adequate "smoothness" in a simple table or even a first difference table, an inordinate number of cards is required.[4] This is a considerable handicap for two reasons: first, a large amount of time is required to initially key punch the table, and second, a large amount of time is required to consult the table after it is in card form.[5]

FIG. 6 GRAPHICAL REPRESENTATION OF THIRD DIFFERENCE TABLE



4 In many problems "smoothness" rather than "accuracy" determines the number of table cards required. The correlations used in technical calculations are often empirical ones which are not too well defined. This fact does not eliminate the necessity in many studies that the approximating function and its first few derivatives vary continuously.

5 The second disadvantage is not too important if a large number of values are to be looked up simultaneously in a table. The problems under discussion, however, require finding only one value in each of several tables for a given problem.

Third difference tables were felt to be the best compromise between ease of construction, speed of use, and both accuracy and "smoothness" of results. Having decided upon third difference tables there remains a choice as to the method of interpolation used in them. The most straight-forward interpolation is done by Newton's formula for forward interpolation. More complicated formulas, such as the central difference formulas of Stirling and Bessel, permit more accurate interpolation in a given difference table. However, the fact that "smoothness" rather than accuracy is generally of controlling importance tends to minimize any advantage to be gained by using these latter equations.

The interpolation formula for third difference tables which has actually been used in this work may be written:

$$y = y_{n-2} + p\ \Delta^1 y_{n-2} + p(p-1)\frac{\Delta^2 y_{n-2}}{2} + p(p-1)(p-2)\frac{\Delta^3 y_{n-2}}{6} \tag{3}$$

where p is defined as:

$$p = \frac{x - x_{n-2}}{\Delta x} \qquad (x_{n-1} \leqq x \leqq x_n) \tag{4}$$[6]

The base value of y and the differences are taken at x_{n-2} since this procedure will give the most accurate value of y for the interval $x_{n-1} \leqq x \leqq x_n$. This is because the differences used in equation (3) were calculated from an equal number of points both before and after x.

Illustrative Example

It is desired to calculate the value of y at x = 837 from the third difference table given in Table 1. First, p is calculated from Eq. (4):

$$p = \frac{837 - 750}{50} = 1.740$$

Then, from Eq. (3):

$$\begin{aligned} y &= .9900 + (1.740)(-.0150) \\ &\quad + \frac{(1.740)(.740)(-.0045)}{2} \\ &\quad + \frac{(1.740)(.740)(-.260)(.0121)}{6} \\ &= .9603 \end{aligned}$$

6. The tabular interval, Δx, must be constant for a given table of this type.

Card Layout for Third Difference Tables

Figure 7 shows a typical IBM table card. Each card carries the following information:

Cols.	Data		
17-19	Code Number	=	identification of a particular table (constant for a given table). All table cards have an X in Col. 17.
20-27	x_n	=	argument value
28-33	y_{n-2}	=	function value
34-39	$\Delta^1 y_{n-2}$	=	first difference
40-45	$\frac{1}{2}\,\Delta^2 y_{n-2}$	=	second difference divided by two
46-51	$\frac{1}{6}\,\Delta^3 y_{n-2}$	=	third difference divided by six
52-59	$\frac{1}{\Delta x}$	=	reciprocal tabular interval for argument (constant for a given table).

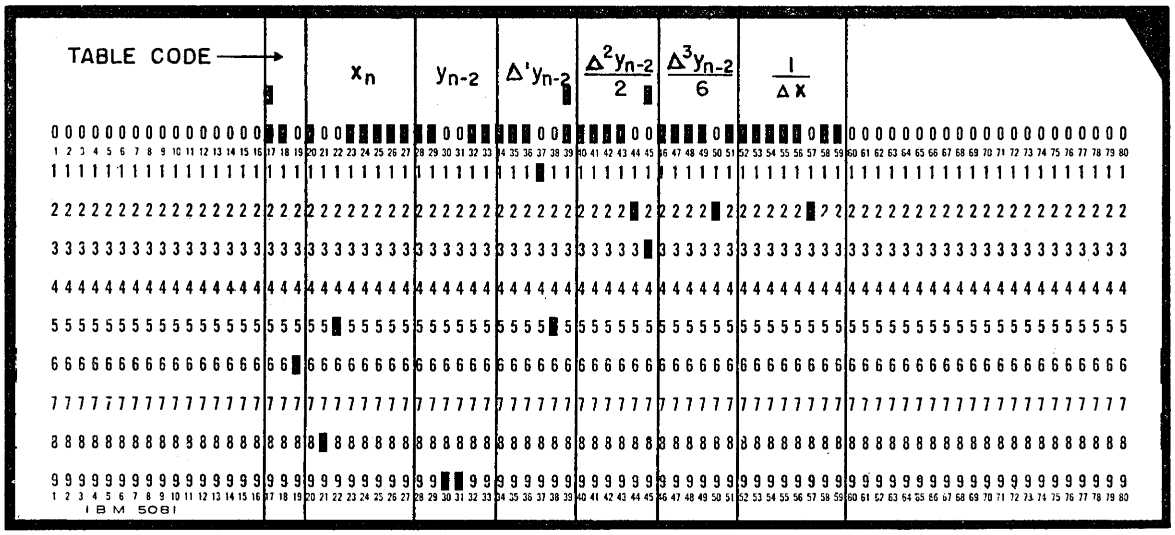


Fig. 7 Table Card

The argument, x, and the reciprocal argument interval, $\frac{1}{\Delta x}$, are eight-digit positive numbers in the form xxxx.xxxx. The function and its differences are six-digit numbers in the form xx.xxxx. They may be positive or negative; if the latter, they carry an X punch in the last position. The choice of the size of the numbers to be used is, of course, to some extent arbitrary. Modifications should not be made, however, without considering whether changes in control panel wiring also become necessary.

Tables will in general be constructed to cover some limited range of an argument. In order to secure protection against inadvertently over-running a table, each table is followed and preceded by a card bearing an X punch in Col. 22. This X is a control punch to stop the 602-A calculator should it receive one of these cards during an interpolation calculation.

The last "X in Col. 22" card for each table has an argument value of 9999.9999. The first "X in Col. 22" card has an argument value which is 0.0001 lower than the lowest value the table covers. If a table starts with an argument value of 0000.0000 it has no low limit "X in Col. 22" card. The fact that these error cards will be selected if the table is over-run should become clear in the next section.

Selection of Table Cards

Each of the problems for which this system was designed involves one table look-up in each of 30-60 tables. To do this all the tables are placed in consecutive order of their Code Numbers (Cols. 17-19) and the cards of each table are in consecutive order with respect to their argument values. This "composite" table deck is then placed in the primary feed of an IBM collator. A set of data cards which carry a table Code Number in Cols. 17-19 and the argument value to be looked-up in Cols. 20-27 is placed in the secondary feed. Each set of data cards must contain only one argument value to be looked up in each individual table.

The collator is wired with the control panel shown in Fig. 12. The result of this control is that each data card selects the appropriate table and the card in that table with the next highest (or equal) argument value. The two cards eject together to the second pocket of the collator, with the table card on the bottom. Unselected table cards are rejected to the Primary Select pocket of the collator.

If a data card exceeds the range of the table, it selects the 9999.9999 card with the X in Col. 22. If the data card is below the prescribed range of the table it selects the low error card. (For example, if the lower limit of a table is 500.0000, then a data card requesting a value at 487.0000 would select an error card 499.9999 carrying an X in Col. 22.)

Interpolation Calculation on the 602-A

The merged stack of table and data cards is taken directly from the collator to the card feed of the 602-A. The 602-A is controlled by the interpolation panel which is described under INTERPOLATION CONTROL PANEL.

In this operation all the necessary interpolating information is read from the selected table card by the 602-A and stored. Then the data card is read, the interpolation calculation is performed, and the answer is punched in Cols. 1-8. If the answer is negative the card is X punched in the units position (Col. 8). The calculator then proceeds to the next pair of cards.

The data cards (punched with answers) are now separated from the table cards (X in Col. 17). The latter are remerged with the tables from which they came.

Simultaneous Running of Several Problems

Ten duplicate sets of "composite" table decks are kept on file, each with a digit punch from 9 to 0 in Col. 69. Each of several sets of data cards to be looked-up in tables is assigned a Problem Number which is punched in Cols. 66-69. The last digit of this Problem Number is then compared with the number in Col. 69 of the table cards during the selection operation on the collator. Thus problems with ten consecutive digits in the last column of the Problem Number can be run together in one machine operation. The 602-A calculator performs the interpolation calculation on paired cards without regard to their Problem Number, and thus can deal with any number of problems in one machine run.

Scope and Speed of Operations

Typical problems to which this method has been applied each involve consulting 30-60 tables, depending upon the nature of the problem. The average third difference table used consists of from 5 to 10 cards. Thirty such tables can be run through the collator in a machine time of about one minute. The interpolation calculation on the 602-A takes about four minutes for thirty interpolations.

COMBINATION PROBLEMS

In catalytic cracking yield and quality predictions it is necessary to perform the final calculations, which are sequential in nature, through card-programming. However, many of the numbers which are used in these calculations are secured from difference tables.

Those cards of the program deck which require tabulated data are key punched in Cols. 20-27 with the argument value at which that data is required. These cards in the program deck are always looked-up in the same tables in each problem, so they already carry in Cols. 17-19 the necessary Table Code Number. Once these cards have been key punched with the argument they simply become data cards which can be run through the interpolation procedure in the normal manner. The interpolated answers from this procedure are punched in Cols. 1-8 by the 602-A so that the cards can then be inserted in the proper order in the program deck.

There are actually three classes of cards in a typical program deck:

> Class A cards which require data to be key punched in Cols. 1-8 for direct use in the card-programmed calculation. These cards are identified by an X in Col. 2 and a typical card is shown in Fig. 8.

Class B cards which require data to be punched in Cols. 20-27 for table look-ups before use in the card-programmed calculation. These cards are identified by an X in Col. 14 and a typical card is shown in Fig. 9.

Class C cards which require no special key punching.

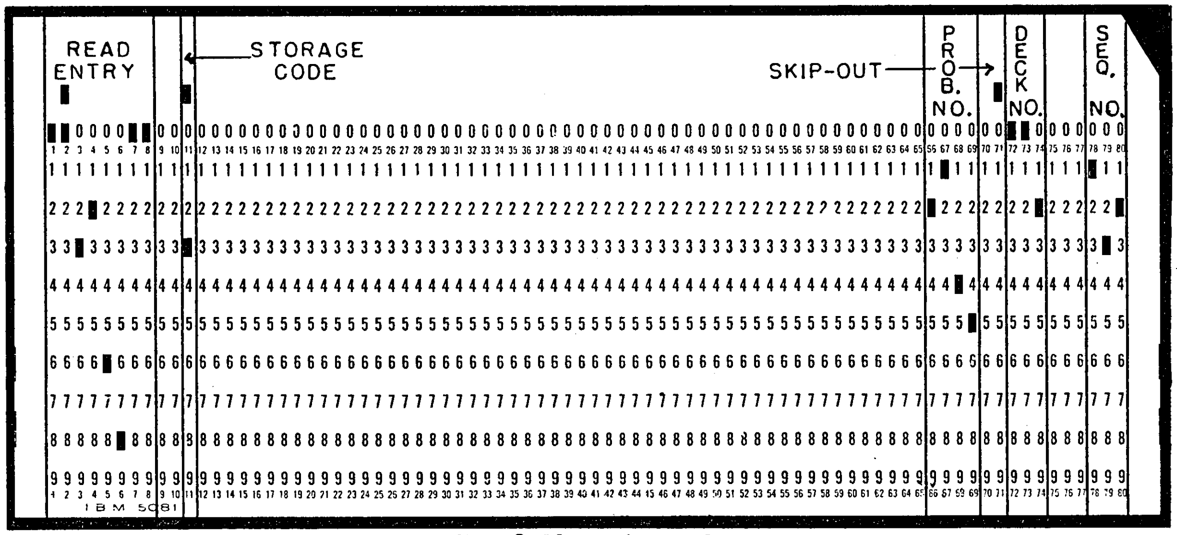


Fig. 8 Class A Card

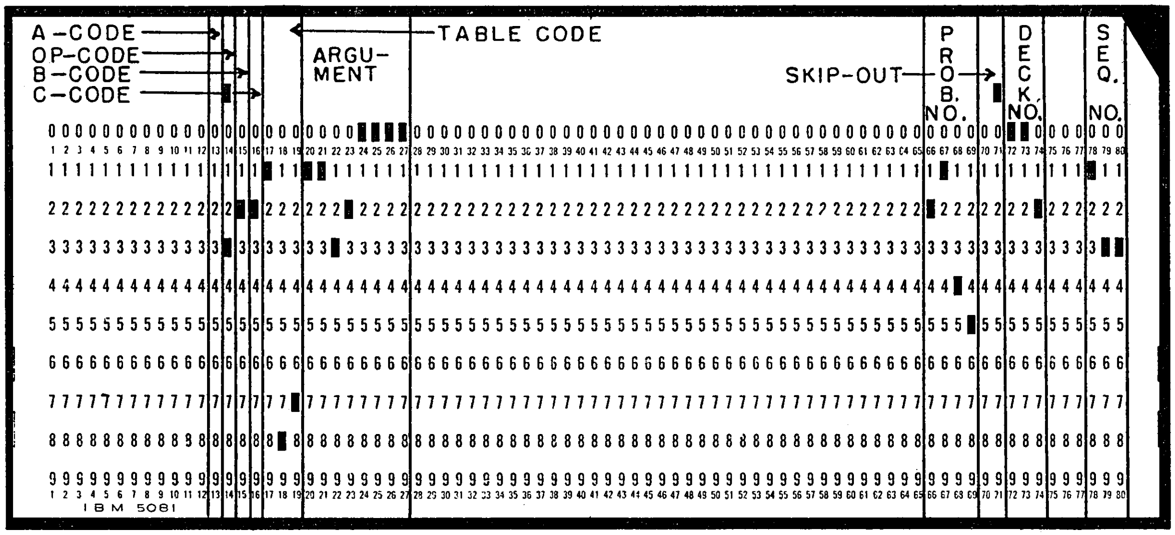


Fig 9 Class B Card

All punches which are invariant from problem to problem, such as identification numbers, control instructions and numerical constants are, of course, pre-punched on the program deck.

It has been found preferable to have the data to be used in Class A and Class B cards key punched into blank cards rather than directly into the program deck. This minimizes handling of the program deck itself. The blank cards must be key punched with Sequence Numbers (in the case of Class A cards) or Table Code Numbers (in the case of Class B cards) for identification. The Class B cards then go through the table look-up procedure and answers are punched in Cols. 1-8. The data in Cols. 1-8 on both Class A and B cards are then transferred to the actual program deck by the IBM Reproducing Punch. The original data and table look-up values are later listed as a matter of record.

Fig. 10 shows a flow chart of the operations which are gone through in the complete solution of a combination problem. Although this may appear somewhat intricate at first, it is actually quite straightforward for an experienced IBM operator and soon becomes a smooth routine. When a group of problems is done simultaneously the intermediate manual operations are done only once for the group, so that the overall time consumed per problem decreases markedly.[7]

CARD-PROGRAMMING CONTROL PANEL

This section presents detailed instructions for wiring the control panel which permits the IBM 602-A calculator to function as an elementary card-programmed calculator. It is a modification of a board described by E. V. Hankam of the Watson Scientific Computing Laboratory.

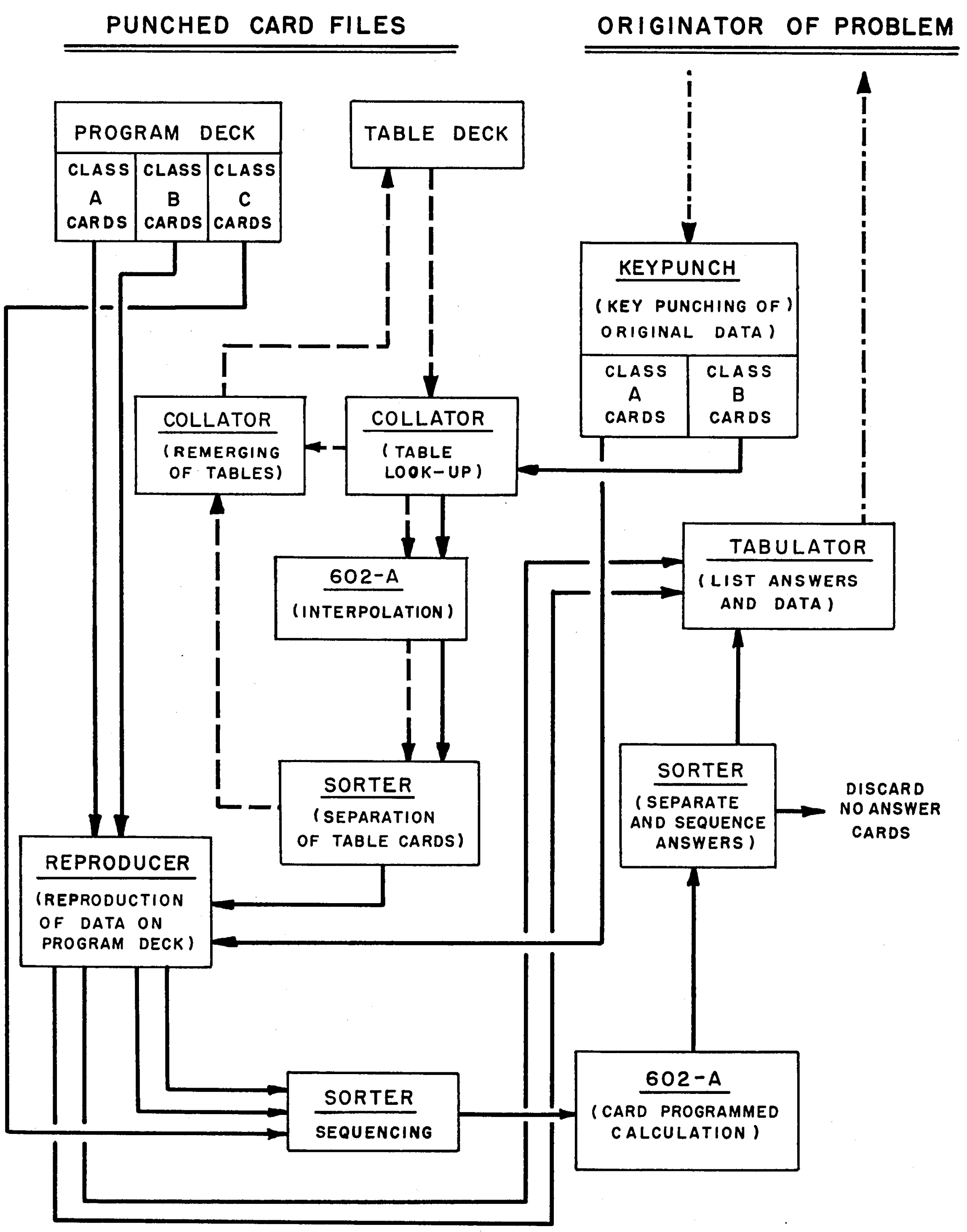
The 602-A calculator to be used with this panel must be an essentially full capacity machine. Eight storage units, eight counters, fourteen pilot selectors and ten co-selectors are used.

Operations — Addition, subtraction, multiplication and division can be performed on eight-digit numbers with algebraic signs on a fixed decimal basis. The calculator has been wired so that numerical data read in the form ±xxxx.xxxx will automatically be developed to eight-digit answers with the same decimal location. No provision has been made for rounding or dropping of insignificant figures. Eight (or less) digit numbers which cannot be expressed in the form xxxx.xxxx can be handled by the machine. This may, however, require special programming and a consequent reduction in machine speed.

Storage — As many as seven eight-digit numbers can be stored in the machine. Negative numbers are stored in complement form, thus freeing pilot selectors for other operations. All numbers are stored in any one of seven storage units (2), (3),...(8).

7. The desirability of eliminating these intermediate operations is, of course, only minimized by this fact; this is a major reason why it is so advantageous to have the Card-Programmed Electronic Calculator.

FIG. 10 FLOW OF OPERATIONS IN COMBINATION PROBLEMS


PUNCHED CARD FILES
ORIGINATOR OF PROBLEM
PROGRAM DECK
CLASS A CARDS
CLASS B CARDS
CLASS C CARDS
TABLE DECK
KEYPUNCH
(KEY PUNCHING OF ORIGINAL DATA)
CLASS A CARDS
CLASS B CARDS
COLLATOR (REMERGING OF TABLES)
COLLATOR (TABLE LOOK-UP)
TABULATOR (LIST ANSWERS AND DATA)
602-A (INTERPOLATION)
SORTER (SEPARATION OF TABLE CARDS)
SORTER (SEPARATE AND SEQUENCE ANSWERS)
DISCARD NO ANSWER CARDS
REPRODUCER (REPRODUCTION OF DATA ON PROGRAM DECK)
SORTER SEQUENCING
602-A (CARD PROGRAMMED CALCULATION)

Data Cards — Numbers may be entered directly from data cards. Negative numbers are identified by a suitable X-punch.

Instruction Cards — Data cards are followed by instruction cards. Each instruction card contains a numerical code to perform an operation of the type A ʘ B = C, i.e., two of the stored factors are selected, an operation is performed on them, and the result of that operation is stored, punched or both. Instruction cards may also carry the number A, instead of obtaining it from storage in the machine.

Notation and Assignment of Units

Brackets enclose counter groups; parentheses enclose storage units.

[I]	=	[1, 2, 3]	MC (multiplicand), DD (dividend)
[II]	=	[5, 6, 7, 8]	Result of operations
[4]	=	[4]	Store C-code
(i)	=	(2), (3),... (8)	Store data and intermediate results
(1R)	=	(1R)	MP (multiplier), DR (divisor)
(1L)	=	(1L)	Store B-code

Flow of Operations

MULT	:	$(i) \xrightarrow{P_1} [II] \xrightarrow{P_2} [I] \xrightarrow{P_3} (1R)$	;	$(i) \xrightarrow{P_4} [II] \xrightarrow{P_5} [I]$	;	$\overset{P_6}{\overline{\text{MPL}}}$	;	$[II] \xrightarrow{P_7} (i)$
DIV	:	$(i) \rightarrow [II] \rightarrow [I] \rightarrow (1R)$	;	$(i) \rightarrow [II] \rightarrow [I]$	;	DIV	;	$[II] \rightarrow (i)$
ADD	:	$(i) \rightarrow [II]$	;	$(i) \rightarrow [II]$			;	$[II] \rightarrow (i)$
SUB	:	$(i) \rightarrow [II]$	;	$(i) \rightarrow [II]$			;	$[II] \rightarrow (i)$

Remarks: Numbers have to be balance tested, and converted if complements, before being used as factors in multiplication and division. The operation A ʘ B = C is performed such that the first factor A is the multiplier, divisor or minuend.

Programming of Data Cards

Card Form:	Col. 1-8	:	$\pm A_i$ (X-punch for minus sign in Col. 8)
	Col. 11	:	storage code (2, 3, . . . 8) ; also X-punch to differentiate from instruction cards
	Col. 71	:	X-punch for SKIP OUT on all cards except those to be punched

Step	Operation	Instructions
R	$\pm A_i \longrightarrow$ [II]	[II] RI ± (sign is wired from control brush to P-Sel 1) Storage code (Col. 11) is wired (via Digit Sel 2) to p.u. P-Sel 2, 3, ... 8 from Read brush X in Col. 11 is wired to P_7 - SKIP from read brush.
P_7	[II] $\longrightarrow$ (i)	[II] RR ; (i) RI ; READ [I] RE ; (resets MC and Remainder on instruction cards)

Remarks: Only one program step is required for data cards. Since this program step is identical with P_7 of instruction cards, P_1 to P_6 are skipped on data cards. Punching on both data and instruction cards takes place whenever a number is read into (8). On other cards an X-punch in Col. 71 is wired to SKIP OUT.

Programming of Instruction Cards (A ⊙ B = C, i.e. $A_i \odot A_j = A_k$)

Card Form:	Col. 1-8	:	numerical data or blank[8]	
	Col. 13	:	A-code or blank (2, 3, ... 8)[8]	
	Col. 14	:	Operation - code (1, 2, 3, 4)	
	Col. 15	:	B-code	(2, 3, ... 8)
	Col. 16	:	C-code	(2, 3, ... 8)
	Col. 71	:	X for SKIP OUT	

Step	Operation	Instructions
R		A-code $\longrightarrow$ via Digit Sel 2 to p.u. P-Sel 2, 3, ... 8 from read brush
		Op-code $\longrightarrow$ via Digit Sel 1 to p.u. P-Sel 11, ... 14 from read brush
	B-code $\longrightarrow$ (1L)	(1L) RI, i.e. (1) RI from read brush
	C-code $\longrightarrow$ [4]	[4] RI+ from read brush
		R.D.O. P-Sel 2, ... 8 through Co-Sel 9 and 10 p.u. at Read Cycle
	$\pm A \longrightarrow$ [II]	[II] RI± (sign is wired from control brush to P-Sel 1)

8 If Col. 13 contains an A-code punch, then Cols. 1-8 must be blank.
If Cols. 1-8 contain data, then Col. 13 must be blank.

Step	Operation	Instructions
P_1	(i) $\longrightarrow$ [II]	(i) RO ; [II] RI+ N. B. test [II] : P-Sel 9 through Co-Sel 2 p. u. at P_1 d. o. P-Sel 2. . . 8 with "9" pulse through Co-Sel 2 (1L) RO to p. u. P-Sel 2. . . 8 through Digit-Sel 2 P_4 -SKIP through P-Sel 11 and 12 (if either is p. u.) through T-side of Co-Sel 2
P_2	[II] $\longrightarrow$ [I]	[II] RR ; [I] RI± (± through P-Sel 9)
P_3	[I] $\longrightarrow$ (1R)	[I] RO ; [I] RE ; (1R) RI
P_4	(i) $\longrightarrow$ [II]	(i) RO ; [II] RI+ (- for subtraction through P-Sel 12) N. B. test [II] : P-Sel 10 through Co-Sel 3 p. u. at P_4 d. o. P-Sel 2. . . 8 with "9" pulse through Co-Sel 3 [4] RR to p. u. P-Sel 2. . . 8 through Digit-Sel 2 P_7 -SKIP through P-Sel 11 and 12 (if either is p. u.) through N-side of Co-Sel 2
P_5	[II] $\longrightarrow$ [I]	[II] RR ; [I] RI± (± through P-Sel 10)
P_6	$A \overset{\times}{\underset{\div}{}} B$	[II] RI±(through P-Sel 9 and 10 and Co-Sel 4) MPL ; [I] RO (T-side of P-Sel 13 or N-side of P-Sel 14) DIV ; (1R) RO ; [I] RI- (T-side of P-Sel 14 or N-side of P-Sel 13)
P_7	[II] $\longrightarrow$ (i)	[II] RR ; (i) RI ; READ [I] RE (resets MC or Remainder)

Remarks: Whenever i = 8, (8) PUNCH is wired together with (8) RI on P_7. An All Cycles impulse is wired to "Punch Interlock" so that punched results can also be read out of (8) on the following instruction card.

Selectors:

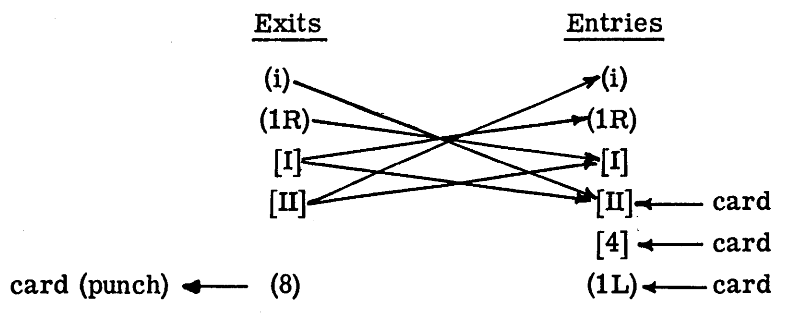
			d. o.
Pilot-Sel:	1	SIGN of A_1 on data cards (control brush)	R. D. O.
	2	a) with A-code from read brush on all cards	end of P_1
	⋮	b) with B-code from beginning of P_2	end of P_4
	8	c) with C-code from beginning of P_5 (through Digit Sel 2 in all cases)	R. D. O. (through Co-Sel 1)

9	N.B. test of [II] to test A at P_1	R.D.O.
10	N.B. test of [II] to test B at P_4	R.D.O.
11	+	
12	-	
13	×	from read brushes (Col. 14) through Digit Sel 1 R.D.O.
14	÷	
15	free	
16	free	
17	free	

Note: P-Sel 1, 9 and 10 are impulsed at the X p.u.
The other P-Sels are impulsed at the digit p.u.

Co-Sel:		
1	:	free
2	:	P_1 couple exit
3	:	P_4 couple exit
4	:	Coupled to P-Sel 10
5 & 6	:	P_7 couple exit through T-side of P-Sel 13
7	:	P_6 (to fill up [II] with "9"s through N-side of Co-Sel 9)
Note:		Highest 4 positions of [II] are bussed
8	:	free
9 & 10	:	read couple
11 & 12	:	coupled to P-Sel 11 and 12

Channels:



Shifting:

A number is read out of [II] in two possible ways:

a) No shift (whenever operation is not MULT)

b) Shift by four (read out of 5th and drop 4) (whenever operation is MULT)

a) is wired through N-side of Co-Sel 5 & 6

b) is wired through T-side of Co-Sel 5 & 6

Note: With the exception of the channel of (1R) into [I] the units position of all numbers are read into or out of the 12th position of [I]. This must be done to line up the DD and, for the sake of uniformity, can be done for all other operations.

General Remarks

1. There is no provision for reading out the remainder, or rounding.
2. (1R) will read in extraneous numbers from data cards; [(1) is impulsed to RI on all cards]; however these numbers will clear out later.
3. By means of multiple punches in the storage code a factor can be read from a data card into several storage units. If it is required to read the result C into two storage units, the C-code cannot have a double punch because it is read into a counter from the card.
4. The A- or B-code can be left blank so that none of the P-Sel 2, 3,...8 will p.u. This has to be done in case of transfer operations $A + O = A$ (or $O \pm B = \pm B$).
5. Instruction cards are blank where data cards are punched, and vice-versa (except for numerical information which may be punched in Cols. 1-8, and X for SKIP OUT in Col. 71).
6. When a number is entered in Columns 1-8 of an instruction card, this number will be read into [II] which receives a RI impulse on every card. (P-Sel 1 controls whether it is RI+ or RI-.) The number entered may of course be $10^3, \ldots, 10^{-3}$, and thus can be used to shift any number held in a storage unit by three columns to the left or right.

CARD PROGRAMMING EXAMPLE

As an extremely simple example of card program planning, consider the following equation:

$$w = 1.4300 + 0.2389\ x - 0.5432\ [\frac{x}{y} - z]$$

Given the three pieces of data, x, y and z, the problem is to evaluate w.

A sample program planning chart is shown in Fig. 11. The first three cards "load" the calculator with data. The following six cards control the calculation and finally punch the answer into the last card. Several of the instruction cards illustrate the use of cards as direct entries for the factor A in an operation $A \odot B = C$.

As mentioned above, this is an extremely simple example. Sequential calculations involving more than two hundred cards have been done and even lengthier calculations are planned. In these longer problems it is frequently necessary to introduce new data cards throughout the program deck, rather than merely "loading" the calculator with the first several cards of the deck.

CARD PROGRAMMED 602-A CALCULATOR PLANNING CHART

1-8	11	13	14	15	16	71	Storage Units								75-77	78-80
Card Entry	St.	A	Op.	B	C	Skip Out	(2)	(3)	(4)	(5)	(6)	(7)	(8)		Ans. No.	Card No.
x	X 2					X	x									001
y	X 3					X	x	y								002
z	X 4					X	x	y	z							003
0.2389			3	2	5	X	x	y	z	.2389x						004
1.4300			1	5	5	X	x	y	z	1.4300 +.2389x						005
		3	4	2	2	X	$\frac{x}{y}$	y	z	1.4300 +.2389x						006
		2	2	4	2	X	$\frac{x}{y} - z$	y	z	1.4300 +.2389x						007
.5432			3	2	2	X	$.5432 [\frac{x}{y} - z]$	y	z	1.4300 +.2389x						008
		5	2	2	8		$.5432 [\frac{x}{y} - z]$	y	z	1.4300 +.2389x			w		001	009

Program Deck No. : 008 Purpose: Illustrative Example Date: By:

Figure 11

COLLATOR WIRING FOR TABLE LOOK-UP

Figure 12 shows the wiring of the collator for the table look-up operation. After this processing the selected table cards, together with their data cards can now be fed directly into the 602-A.

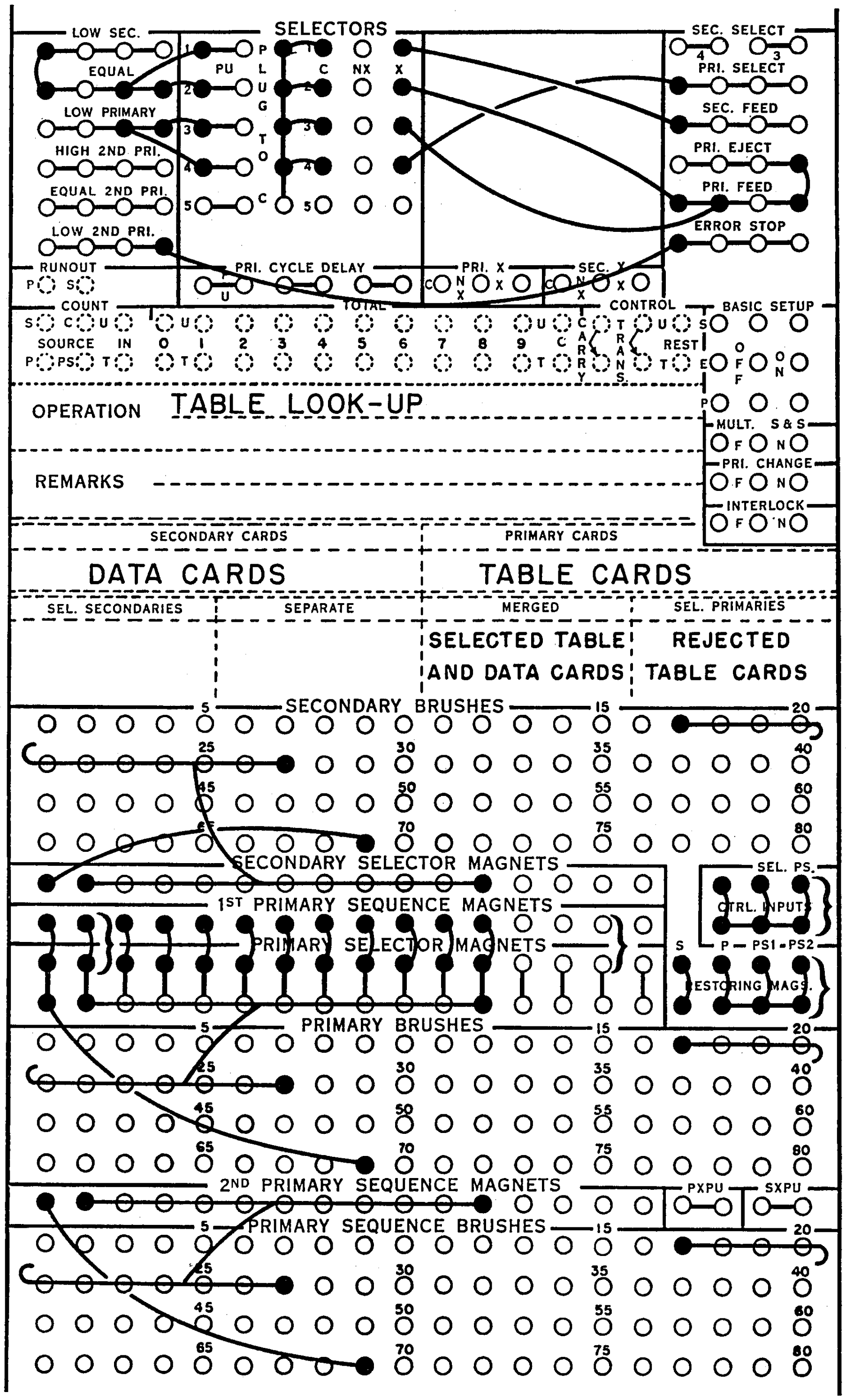


FIGURE 12

WIRING OF COLLATOR FOR TABLE LOOK-UPS

INTERPOLATION CONTROL PANEL

The instructions given below, together with Figure 13, are complete wiring directions for the interpolation calculation discussed earlier. The wiring diagram is only for showing the wiring of channels between storage units, counters, and reading brushes or emitters. It is shown since the use of co-selectors, to prevent back circuits, is somewhat intricate. Control wiring is described in sufficient detail below, and punch wiring (including balance test and conversion) is routine.

The wiring described here gives complete sign control for positive and negative values of the tabulated function and its differences. However, this sign control only holds when the value of x lies in the range $x_{n-1} \leqq x \leqq x_n$ where x_n is the first tabulated value of the argument larger than x. (This x_n table card is selected by an x data card when the collator is wired as shown in Figure 12.)

To use this control panel the 602-A requires eight counters, seven storage units, seven pilot selectors and twelve co-selectors.

CONTROL INSTRUCTIONS - INTERPOLATION PROBLEM

Step	Operation		Instructions
Read X	$x_2 \rightarrow [34]$	(Cols 20-27)	p.u. P-Sel 1 by X in Col 17 (control brush)
	$y_o \rightarrow (2)$	(Cols 28-33)	p.u. P-Sel 3 by X in Col 39 (read brush)
	$\Delta^1 y_o \rightarrow (3)$	(Cols 34-39)	p.u. P-Sel 4 by X in Col 45 (read brush)
	$(\frac{1}{2})\Delta^2 y_o \rightarrow (4)$	(Cols 40-45)	p.u. P-Sel 5 by X in Col 51 (read brush)
	$(\frac{1}{6})\Delta^3 y_o \rightarrow (7)$	(Cols 46-51)	p.u. P-Sel 6 by X in Col 33 (read brush)
	$\frac{1}{\Delta x} \rightarrow (1)$	(Cols 52-59)	RDO P-Sel 1
			(1) (2) (3) (4) (7) RI thru T side Co-Sel 9
			[34] RI+ thru T side of P-Sel 1
			Co-Sel 9 is p.u. by couple exit of P-Sel 1
			[12], [78]RE READ & SKIP OUT
Read NX	$-x \rightarrow [34]$		[34]RI- thru N side of P-Sel 1
			[12], [78]RE
P-1	$\frac{1}{\Delta x}(x_2 - x) \rightarrow [12]$		p.u. Co-Sel 11 & 12
			[12]RI+ [34]RO MPL

Step	Operation	Instructions
P-2	$2-p \rightarrow (1)$	p.u. Co-Sel 2 (1)RI [12]RO
P-3	$(2-p)\ \frac{\Delta^3 y_o}{6} \rightarrow [78]$	(7)RO [78] RI+ MPL
P-4	$(2-p)\ \frac{\Delta^3 y_o}{6} \rightarrow (7)$	p.u. Co-Sel 10 [78] RO (7) RI [34] RI+ [12] RE
P-5	$-(2-p) \rightarrow [34]$	[34] RI- (1) RO [78] RE
P-6	$p \rightarrow (1)$ $y_o \rightarrow [12]$	[34] RO (1) RI [12] RI± through P-Sel 6 $\begin{smallmatrix}N\\T\end{smallmatrix}$ (2) RO p.u. Co-Sel 8
P-7 & 8	$p\,\Delta y_o \rightarrow [12]$ $p\frac{\Delta^2 y_o}{2} \rightarrow [56]$ $p(2-p)\ \frac{\Delta^3 y_o}{6} \rightarrow [78]$	12 RI± through P-Sel 3 $\begin{smallmatrix}N\\T\end{smallmatrix}$ (3)RO (4)RO [56] RI+ (7) RO [78] RI+ MPL
P-9	$-1 \rightarrow [34]$ $p\ \frac{\Delta^2 y_o}{2} \rightarrow (6)$ $p(2-p)\ \frac{\Delta^3 y}{6} \rightarrow (7)$	Imm. p.u. P-Sel 7 [34] RI- [56] RO (6) RI [78] RO (7) RI
P-10	$p-1 \rightarrow (1)$	p.u. Co-Sel 1 [34] RO (1) RI [56] RE [78] RE

Step	Operation	Instructions
P-11 & 12	$p(p-1)\ \frac{\Delta^2 y_o}{2} \longrightarrow [12]$ $p(p-1)\ (2-p)\ \frac{\Delta^3 y_o}{6} \longrightarrow [78]$	12 RI± through P-Sel 4 $\overset{N}{T}$ (6) RO (7) RO [78] RI+ MPL p. u. P-Sel 2 (and Co-Sel 5, 6, 7 from cpl exit) (used for program extension)
P-13	$p(p-1)\ (p-2)\ \frac{\Delta^3 y_o}{6} \longrightarrow [12]$	p. u. Co-Sel 3 & 4 [12] RI± through P-Sel 5 $\overset{T}{N}$ [78] RO [34] RE
P-14 & 15	Punch answer	[12] RO (8) RI, punch DO P-Sel 3, 4, 5, 6 READ (Balance test & convert answer)

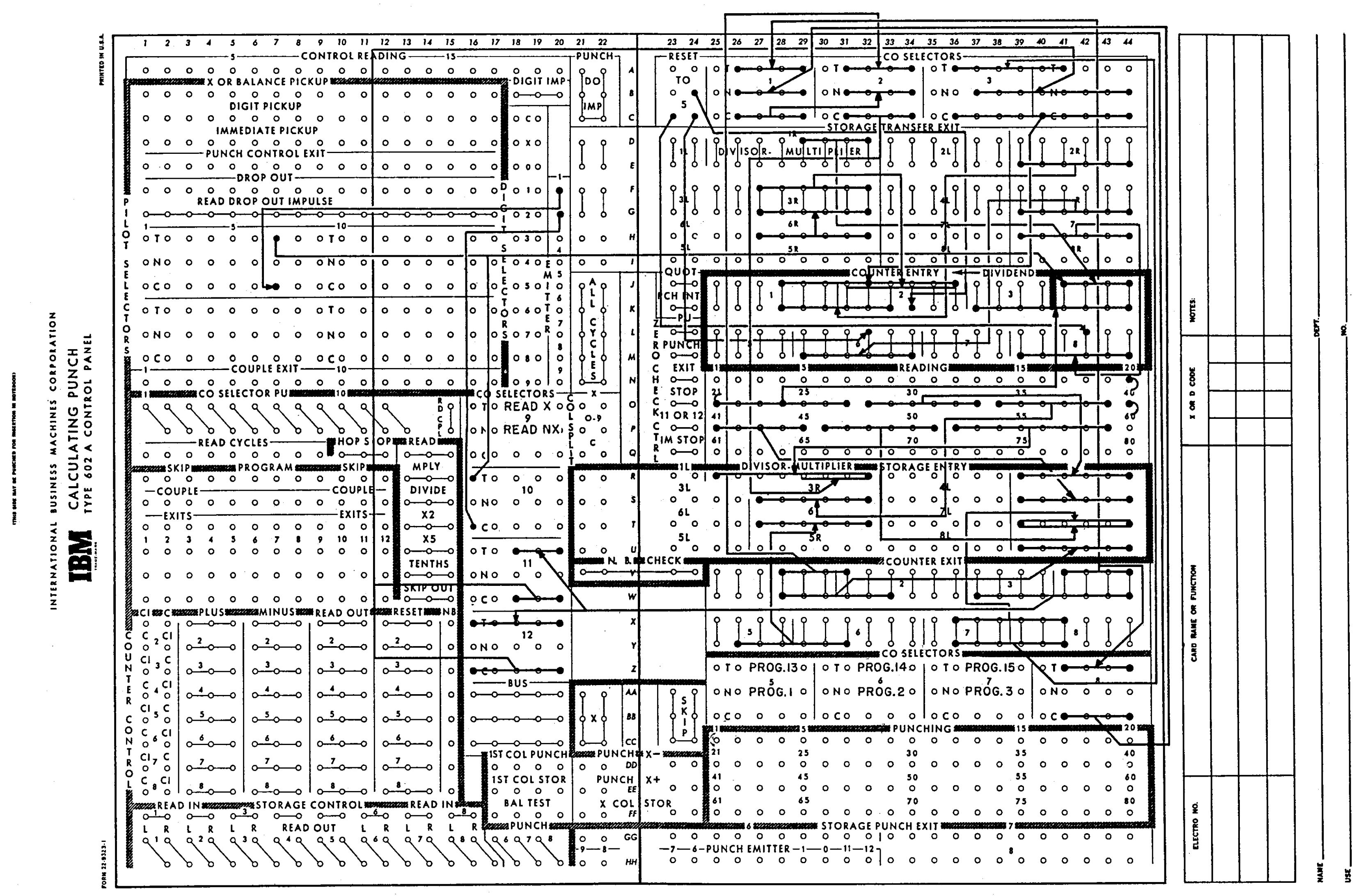


FIGURE 13

WIRING OF STORAGE UNIT AND COUNTER ENTRIES AND EXITS FOR INTERPOLATION

13 DIGIT FLOATING DECIMAL — MODEL II CPC

Dura W. Sweeney
Los Alamos, New Mexico

Editor's Note: This article describes a general purpose 13 significant digit floating decimal setup. Complete planning charts for the 605 follow the article. Preceding these charts are notations on planning for the 418 control panel.

If a number is represented as $n \cdot 10^p$, n given to thirteen digits or less and $|p| < 50$, the number is read into the CPC Model II as n and 50 + p with the sign of n punched as an X over the last digit of (50 + p). For example, $\frac{1}{e} = 3.678794411714 \times 10^{-1}$, the number read into the CPC would be 367879441171449 and -e as $27182818284595\overset{x}{0}$.

Card Layout:

Card Column	
1-4	Serial number
5	Spread read-in control
6-8	A address
9	Operation control
10-12	B address
13-14	C address
6-20	Ctr grp #1 on SRI
21-35	Ctr grp #2 on SRI or channel A card read-in
36-50	Ctr grp #3 on SRI or channel B card read-in
51-65	Ctr grp #4 on SRI
66-80	Ctr grp #5 on SRI or compare number with channel C

Spread Read-In Controls:

The card preceding one in which there are to be numbers read into the five counter groups must contain:

(1) 9 punched in column 5 signals SRI.

(2) Ctr addresses, 1,2,3,4,5 punched in column 5. If selective SRI is desired the absence of any of the numbers 1-5 will prevent the number in the card from disturbing the contents of that counter group.

Counter Controls:

There are five counter groups in the 418:

Ctr. grp.	Address	418 Counters
#1	91	2A, 2B, 6A, 6B
#2	92	2C, 2D, 6C, 6D
#3	93	4A, 4B, 4C, 4D
#4	94	8A, 8B
#5	95	8C, 8D

The counter groups are controlled the same as storage registers in that a number remains in the counter group and is not reset on any read-out, but reading into the counter group resets the counters to zero before read-in. Numbers read into the counter groups are available with a one card delay. Storage registers have the normal two card delay. There is no provision for adding numbers in the counter. The counter groups may be addressed on channels A, B, or C independently of one another.

Storage Control:

It is necessary to use two storage registers to store a thirteen digit number and the two digit exponent. To read a number into a storage bank give a C address to a particular storage register in the bank. This will store the eight high order digits, the two digit exponent, and the sign. The following card should have a 6 punched in column 9 and a C address to another storage register in the same bank. This will store the five low order digits, and again the exponent and sign. It will be noted that A and B addresses consist of three digits. Ordinarily the two high order digits are used for card read-in, counter group, or C $\longrightarrow$A (or B) addresses. The third digit is used for storage control. To call a number from a storage bank on channel A or B a three digit address is used. For example, a 256 in channel A will read-out of storage registers 25 and 26 in the following manner. The eight high order digits of 25 will become the eight high order digits of the thirteen digit number, the two low order digits will become the exponent, and the sign will come from 25, the eight high order digits of 26 (including the first three which are zeros) will become the five low order digits of the thirteen digit number.

If the contents of 25 are 52718281850 and the contents of 26 are 50002845950, then this will read-out on an A address of 256 as: 271828182845950 with the 5 in the high order position of 25 becoming the sign.

It is only possible to read-out one such number from storage. It can be read-out on A and/or B, but two thirteen digit numbers may not be read-out of storage on the same card. It is possible to read-out of only one storage register rather than two. For example, if the constant 900000000000050 is read into register 23 only it will appear there as 9000000050 and if it is addressed as 23 in the high order digits of A and/or B address it will come out as 90000000.....50, the blank spaces will read into the 605 as zeros.

Spread Read-Out Control:

It is possible to read-out of the five counters, print, and punch the numbers contained therein by punching a 9 in column 14. Follow this card by a blank card. It is possible to read-out and print, without punching by the above address and putting Switch #2 in A position. Either operation does not disturb the numbers in the counters.

Compare Channel C:

This operation is mainly for test decks. An X punch in column 25 allows the channel C result of the previous card to be compared with the numbers in 66-80 of this card. If they do not compare the machine will stop. Switch #3 in A position does not allow comparing even if there is an X in column 25.

Octal Conversion:

The operations punch 8 in column 9 will convert the thirteen place floating point decimal number in A to a thirteen place floating point octal number. The exponent read-out will be in true figures. Therefore, the number read-in on A must be positive as the sign of C in this operation is reserved for the exponent. For example, the number $\frac{2}{8} + \frac{1}{8^2} + \frac{9}{8^3} + \frac{5}{8^4} + \frac{3}{8^5}$ reads-in on A as 284515380859449 as a decimal number. Its octal equivalent will read-out as $2215300000000\underline{01}$- or $e^{5.5}$ reads-in as 244691932264252 and its octal equivalent is $3645422117103\underline{03}$. The exponent as well as the number is converted to octal numbers.

Switches:

Set-up Switch #1 on	List every card during calculation
Set-up Switch #2 on	Double space before printing
Switch #1 A position	Used in conjunction with Set-up Switch #1, lists every card, but does not calculate.
Switch #2 A position	Does not allow punching
Switch #3 A position	Does not allow comparing

X and Y Punch Controls:

Col. 1	Y	List this card
Col. 1	X	List the next card
Col. 6	X	$\|A\|$
Col. 7	X	-A
Col. 8	X	Double space before printing
Col. 9		See operations controls
Col. 10	X	$\|B\|$
Col. 11	X	-B
Col. 12	X	Eject paper to next sheet
Col. 14	X	Stop machine if C of this card is negative
Col. 25	X	Compare 66-80 of this card with C of preceding card
Col. 20	X	Number in 6-20 is negative
Col. 35	X	Number in 21-35 is negative
Col. 50	X	Number in 36-50 is negative
Col. 65	X	Number in 51-65 is negative
Col. 80	X	Number in 66-80 is negative

605 Operations

Code in Col. 9	Result
Blank	Transfer A to C without performing other operations
1	A + B
3	A · B
4	A ÷ B
5	$\sqrt{A}$, Read-in B as 999999990002500
6	Split number for storage (See Storage Controls)
8	Octal conversion (See Write-up)

Special Controls	
2	Do not round on multiply or add
7	Do not shift left if zeros are present in high order position of C result
9	Convert exponent of A to floating decimal number
Y	Replace exponent of A by B exponent
X	Add two high order digits of B to A exponent. High order digits of B in the form 50 + p, where p is the change desired in A exponent.

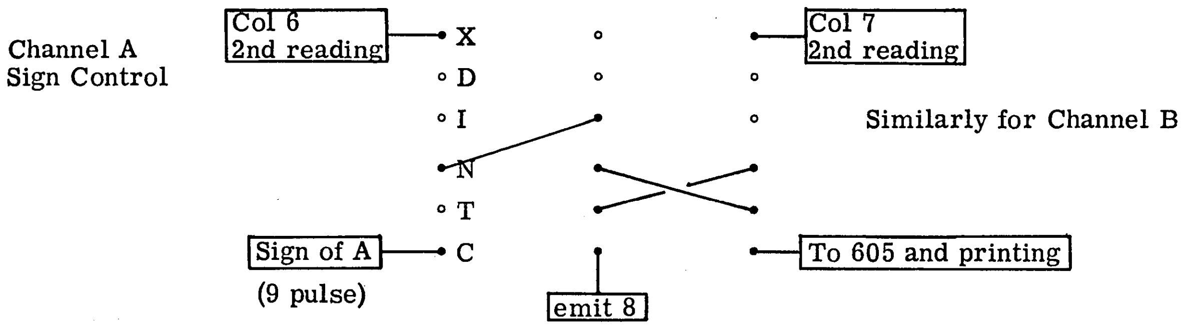
The special controls are used mainly in reduction of arguments for ease of series calculations. Operation code 7 is used for conversion of floating to fixed decimal numbers.

Notations for Planning 418 Panels

The 605 panel, as designed, will fit any standard 605 without additional equipment or modifications. The 605 must be made to operate, however, as if the tubes and units at I-5-U and I-7-T were removed.* The 418 panel as designed at this installation requires an additional Field Selector, therefore the 418 wiring diagrams are not included. It would seem possible to design a 418 panel for a standard machine which will perform all the manipulations described in the write-up except possibly Spread Read-In and Spread Read-Out.

Channel A:	Read the digits into FS 1&2, FS 3&4 (five low order positions) Read the exponent into the two low order positions of the MQ
Channel B:	Read the digits into GS 1&2, GS 4 Read the exponent into the two high order positions of the MQ
Channel C:	Read the digits (R&R) out of the counter exit; Read the exponent out of the two low order positions of GS 3, except on square root. On square root pick-up a selector to read the exponent out of the two high order positions of GS 3.
Operation Code X:	Pick up a selector to read the two high order digits of B into the B exponent position of the MQ.
Operation Code Y:	Pick up a selector to read the B exponent into the A exponent position of the MQ.
Operation Code 7:	Pick up a selector to "emit 5" into Channel Shift
Operation Code 8:	Wire sign hub of GS 3 to counter sign hub to give proper sign to exponent on octal conversion.

There is no provision in the 605 for subtraction. This is done in the 418 by changing the sign before reading into the 605. The following method will enable one to get the negative, absolute, or negative absolute value of A or B. Assume all sign controls are converted to "9 pulses". Then three pilot selectors are wired as follows:



* This can be handled by special arrangement with the IBM Corporation

PANEL A.

	READ OUT	READ IN	SHIFT	PROGRAM		COMMENT
	EXIT 1	EXIT 2	EXIT 3	SUPP	PU	
1	FS1 &2	$\frac{1}{2}$add		①		
2	GS1 &2	ZT	BTSS, Reset	①	all	
3	MQ*	ctr+	**			*GS4ROⓈ ** in 6Ⓢ; Prog Rpt, Rpt Del PU⊕; in 3 $\textcircled{E_x}$
4	RO*	GS3	out 4 **	②		*R&R $\textcircled{\surd}$ **not on $\textcircled{\surd}$
5	emit 5	ctr+ *	in 2 **	㉑	all	*ctr-⊗, $\textcircled{E_x}$; ctr-, BTSS⑧; MQRI $\textcircled{\surd}$ ** in 5 $\textcircled{\surd}$; in 4 $\textcircled{E_x}$
6	GS3 *	ctr- **	in 4 ***	②		*R&R $\textcircled{E_x}$ ** × $\textcircled{\surd}$; MQRI $\textcircled{E_x}$ ***not on $\textcircled{\surd}$, $\textcircled{E_x}$
7	GS3 *	ctr- **	***	②		*GS4RO $\textcircled{\surd}$; RO⑧; emit 5 $\textcircled{E_x}$ **ctr +⊗, $\textcircled{\surd}$, $\textcircled{E_x}$; GS3RI⑧*** in 6Ⓣ; BTSS⊕; in 2 $\textcircled{E_x}$
8	emit 1 *	ctr+ **		④	all	*MQRO⊕, Ⓣ** GS3RI⊕, Ⓣ; ctr-⊕
9	R&R	GS3 *	**	②		*MQRI⊕, Ⓣ; GS4RI⑧ ** out 5 $\textcircled{\surd}$; RS#5PU⊕
10	GS3 *	MQ **	in 5	⑥	⑧$\textcircled{\surd}$	*MQRO $\textcircled{E_x}$** FS1 &2 RI $\textcircled{E_x}$
11	FS1 &2 *	ctr+ **	***	⑤	$\textcircled{E_x}$	*FS3 &4RO $\textcircled{\surd}$** GS4RI $\textcircled{\surd}$*** in 6 $\textcircled{E_x}$
12	MQ*	ctr-, BTSS	Reset	⑳		*GS1 &2 RO⑧
13	FS1 &2	ctr+	in 6	P	⊕$\textcircled{\surd}$	
14	FS1 &2	ctr+	in 5	M	⑧	
15	GS1 &2	+				
16	R&R	FS1 &2			⊕$\textcircled{\surd}$	
17	FS1 &2	ctr+	in 6		⑧	
18	FS3 &4	ctr+	in 6	P		
19	FS3 &4	ctr+	in 5	M	⊕$\textcircled{\surd}$	
20	MQ	FS1 &2	in 4			
21	GS1 &2	+			⑧	
22	R&R	FS3 &4			⊕$\textcircled{\surd}$	
23	FS3 &4	ctr+	in 6			
24	MQ	FS3 &4			⑧	
25	GS1 &2	+	Reset		⊕$\textcircled{\surd}$	
26	MQ	ctr+			⑧	
27	emit 5	$\frac{1}{2}$add	in 2			
28	R&R	MQ	out 3		⊕$\textcircled{\surd}$	
29	MQ	ctr+			⑧	
30	FS3 &4	ctr+	in 4			
31	FS1 &2	ctr+	in 6		⊕$\textcircled{\surd}$	
32	GS4	MQ			⑧	
33	RO	GS4*				*MQRI⑧
34	R&R	FS1 &2	out 6		⊕$\textcircled{\surd}$	
35	FS1 &2	×		①	⑧	
36	R&R*	FS3 &4	out 6	⑦		*MQRO⑧
37	FS3 &4	ctr+	in 3	①	⊕$\textcircled{\surd}$	
38	GS1 &2	+	Prog Rpt	①		
39	R&R	FS3 &4		⑦		
40	FS3 &4	ctr+	in 6	⑦	⊕$\textcircled{\surd}$	
41	MQ	FS3 &4	Rpt Del PU*			*Rpt Del DO(RS7T)
42	GS1 &2	+	Reset			
43	MQ	ctr+			⊕$\textcircled{\surd}$	
44	FS3 &4	ctr+	in 6			
45	emit 5	$\frac{1}{2}$ add	in 2			
46	R&R	FS3 &4	out 3		⊕$\textcircled{\surd}$	
47	FS1 &2	ctr+	in 6	①		
48	GS4	ctr+	RS #6 PU	①		
49	FS3 &4	ctr-		①	⊕$\textcircled{\surd}$	
50	emit 5	MQ	in 5	⑧		
51	GS4	×		⑨		
52	R&R	GS4	out 6	⑨		
53	GS1 &2	×	Prog Rpt *		$\textcircled{\surd}$	*not(RS8T)
54	FS3 &4*	×	RS # 7 PU			*FS1 &2 RO(RS8T)
55	R&R	FS3 &4	out 6	⑦		
56	FS3 &4	ctr+		⑦	$\textcircled{\surd}$	
57	GS1 &2	ctr-, BTSS	Reset	⑦		
58	FS3 &4	GS1 &2		P		
59	GS4	FS3 &4 *	RS#8 &9 PU	⑲	$\textcircled{\surd}$	*ctr+(RS8T)
60	RO	GS4	Rpt Del PU	⑲		

PANEL B.

	READ OUT	READ IN	SHIFT	PROGRAM		COMMENT
	EXIT 1	EXIT 2	EXIT 3	SUPP	PU	
1	emit 8	MQ	Rpt Del DO			
2	GS4	×			⑧	
3	RO	GS4				
4	R&R	MQ	out 6			
5	MQ	ctr+			⑧	
6	emit 8	MQ				
7	FS1 &2	×				
8	RO	FS1 &2			⑧	
9	FS1 &2	ctr-				
10	FS3 &4	ctr+	in 2 *		⊗⊕	*BTSPU #1 Ⓣ
11	GS4	FS3 &4	in 2	⑩	Ⓣ	
12	R&R *	GS4 **	out 2 ***	M		*RO Ⓣ **MQRI ⊗ ***not on Ⓣ
13	FS1 &2	ctr+	in 6	⑩	⊗⊕	
14	GS1 &2	FS1 &2 *		M	Ⓣ	* × ⊗
15	R&R *	GS1 &2	out 6	⑩		*RO Ⓣ
16	R&R	FS3 &4	*	⑪	⊗⊕	*out 6 ⊗
17	GS4*	MQ**			Ⓣ	*MQRO ⊕ ** $\frac{1}{2}$ add ⊕
18	GS1 &2*	ctr+ **	in 3 ***	⑱		*emit 8 ⊕ **ctr-, BTSS ⊕ ***not on ⊕
19	RO	GS4	out 6	③	⊗⊕	
20	GS4	ctr-	in 6	③	Ⓣ	
21	R&R	GS1 &2	out 3	③		
22	FS1 &2	×		③	⊗⊕	
23	GS1 &2	MQ		③	Ⓣ	
24	R&R	GS1 &2	out 6	③		
25	FS1 &2	×		③	⊗⊕	
26	GS1 &2	ctr+		③	Ⓣ	
27	FS3 &4	ctr+		③		
28	FS3 &4	ctr+	in 6			
29	R&R	FS3 &4	out 6		⑧	
30	FS3 &4	ctr+				
31	R&R	FS3 &4	out 4			
32	FS3 &4	ctr+			⑧	
33	GS1 &2	ctr+	in 6			
34	RO	GS4	GS #4 DO	M		
35	RO	GS1 &2	out 5	P	⑧	
36	RO	MQ	Prog Rpt	P		
37	emit 1	ZT	Reset	P		
38	GS3	ctr+	GS #4 PU	P	⑧	
39	emit 1	ctr-		P, NZ		
40	R&R	GS3		P		
41	emit 1	FS3 &4	in 6	P	⑧	
42	GS1 &2	ctr+		P		
43	FS3 &4	ctr-, BTSS	in 3, Reset	P		
44	MQ	FS3 &4	in 5		⑧	
45	R&R	FS1 &2	out 6	GS 4		
46	Prog Rpt, Rpt Del PU	RS #1 PU		㉒		
47	Prog Rpt, Rpt Del PU	RS #2 PU		㉓	⑧	
48	emit 8	MQ	in 5	㉒		
49	emit 1 *	ctr+	in 6 **	㉔		*GS3RO $\textcircled{\text{RS3T}}$ ** not on $\textcircled{\text{RS3T}}$
50	emit 8	+	Reset *	㉔	⑧	* not on $\textcircled{\text{RS3T}}$
51	MQ	GS1 &2 *	in 4 **	GS 4		*ctr+ $\textcircled{\text{RS3T}}$ ** in 2 $\textcircled{\text{RS3T}}$
52	GS3	ctr+	Prog Rpt	⑫		
53	emit 1	$\frac{1}{2}$ add		⑯	⑧	
54	R&R	GS3	Rpt Del DO	GS 4		
55	GS4	$\frac{1}{2}$ add	GS4			
56	emit 1 *	ctr-, BTSS **	***	⑰	⑧	*FS1 &2 RO $\textcircled{\text{RS3T}}$ ** ctr+ $\textcircled{\text{RS3T}}$ *** in 6 $\textcircled{\text{RS3T}}$
57	R&R	GS 4	BTSS, Reset	⑫		
58	FS3 &4	GS 4	Prog Rpt, Rpt Del PU	㉓		
59	RS #3, 4, &10 PU	FS3 &4	GS1 &2 RI	㉓	⑧	
60	emit 1	ctr-, BTSS	Reset	㉓		

PANEL C.

	READ OUT	READ IN	SHIFT	PROGRAM		COMMENT
	EXIT 1	EXIT 2	EXIT 3	SUPP	PU	
1	emit 8	ctr+	GS #3 PU	P		
2	emit 4	ctr-, BTSS			⊕	
3	emit 4	ctr+	GS #1 PU	P		
4	emit 2	ctr-, BTSS				
5	emit 2	ctr+	GS #2 PU	P	⊕	
6	emit 1	ctr-, BTSS	Reset			
7	FS1 &2	ctr+		GS3		
8	R&R	FS3 &4	out 3	GS3	⊕	
9		FS1 &2		GS3		
10	FS1 &2	ctr+	in 2	GS1		
11	RO	FS1 &2	out 6	GS1	⊕	
12	R&R	MQ		GS1		
13	FS3 &4	ctr+		GS1		
14	MQ	ctr+	in 6	GS1	⊕	
15	R&R	FS3 &4	out 5	GS1		
16	FS1 &2	ctr+	in 4	GS2		
17	RO	FS1 &2	out 6	GS2	⊕	
18	R&R	MQ		GS2		
19	FS3 &4	ctr+		GS2		
20	MQ	ctr+	in 4	GS2	⊕	
21	R&R	FS3 &4	out 3	GS2		
22	FS1 &2	ctr+	in 5	M		
23	RO	FS1 &2	out 6	M	⊕	
24	R&R	MQ		M		
25	FS3 &4	ctr+		M		
26	MQ	ctr+	in 3	M	⊕	
27	FS3 &4	ctr+	in 2	P		
28	GS4	MQ *	**		⊗⊕	*ctr+⊕ in 3⊕
29	RO	GS1 &2	out 6	③		
30	GS1 &2	ctr-	in 6	③		
31	emit 5	½ add	in 2	⑥	⊗⊕	
32	RO	FS3 &4	out 3			
33	emit 5	½ add	in 2	⑥		
34	emit 4	½ add	in 3	⑥	⊗⊕	
35	R&R	GS4	out 4			
36	FS1 &2	× *	**			*ctr+⊕ **in 5⊕
37	GS1 &2	ctr+	in 3 *		⊗⊕	*in 5⊕
38	GS4	ctr+				
39	GS4	MQ				
40	RO	FS1 &2	out 6			
41	R&R	GS4				
42	emit 1	GS1 &2	in 6			
43	FS1 &2	½ add				
44	GS1 &2	ctr-, BTSS	in 3, Reset			
45	GS3	ctr+				
46	emit 1	ctr+		⑬		
47	emit 1	ctr-		P		
48	R&R	GS3				
49	MQ	ctr-	in 2	⑭		
50	FS3 &4	ctr+	*	⑭		*Reset E_x
51	GS4	ctr+	in 2	P		
52	FS1 &2	FS3 &4	in 2	P		
53	FS3 &4	ctr+	in 6	P		
54	RO	FS1 &2	out 6	P		
55	RO	GS4		P		
56	emit 1	ZT	Reset	P		
57	Prog Rpt	Rpt Del PU*	RS #6 PU	⑮		*Rpt Del DO RS6T
58		GS3		P, NZ		
59	FS1 &2	ctr+	in 6			
60	GS4	ctr+	BTSPU #1			

CALCULATE SELECTORS

1	Ⓐ	● ctr -	● ctr +	● GS3RI	●	●
		● ctr +	● ctr -	● MQRI	● ③, SF#1	●
		● CS4-4N	● CS3-3N	● CS13-3T	● ㉑	●
2	$\textcircled{E_x}$	● Reset	● Sup	● MQRO	● in 6	● FS1&2RI
		●	●	● GS3RO	●	● MQRI
		● $50C_3$	● ⑳	● $10A_1$	● $11A_3$	● $11A_2$
3	$\textcircled{E_x}$	● RS6-4C, CS13-5T CS16-5T, CS18-5T	● in 2	● ctr +	● emit 5	●
		●	● CS10-2C	● CS1-2C	● GS3RO	● in 4
		● PSF#9	● CS8-4N	● CS6-3N	● CS6-2N	● CS8-1N
4	$\textcircled{E_x}$	● MQRI	● R&R	● in 4	● ctr -	● in 3
		● ctr -	● GS3RO	● in 2	● CS1-1C	● CS10-1C
		● CS7-5N	● $6A_1$	● CS7-4N	● CS6-1N	● CS15-2N
5	$\textcircled{NR}$ ⑧	● MQRI	● MQRO	● Sup	●	● GS1&2RO
		● GS4RI	● R&R	●	●	● MQRO
		● $33A_2$	● $36A_1$	● ⑥	●	● $12A_1$
6	⑧	● ctr-, BTSS	● RO	● GS3RI	● GS4RI	● Prog Source
		● CS4-4C	● CS3-4C	● CS3-3C	● CS13-3C	●
		● CS7-3N	● CS8-2N	● CS8-3N	● $9A_2$	● RS1-3N $\boxed{55\text{-}60B}$
7	$\textcircled{\surd}$	● R&R	●	● MQRI	● in 5	● ×
		● RO	● out 4	● CS6-1C	● CS4-3C	● CS4-1C
		● $4A_1$	● $4A_3$	● CS17-1N	● $5A_3$	● $6A_2$
8	$\textcircled{\surd}$	●	● GS4RO	● ctr +	● in 6	● out 5
		● CS3-5C	● CS6-2C	● CS6-3C	● CS3-2C	● CS10-3C
		● $6A_3$	● $7A_1$	● CS17-2N	● $7A_3$	● $9A_3$
9	$\textcircled{\surd}$	●	● Prog Source	● FS3&4RO	● GS4RI	●
		●	●	● FS1&2RO	● ctr +	●
		●	● $\boxed{52\text{-}60A}$ PSF#3, PSF#4	● $11A_1$	● $11A_2$	●

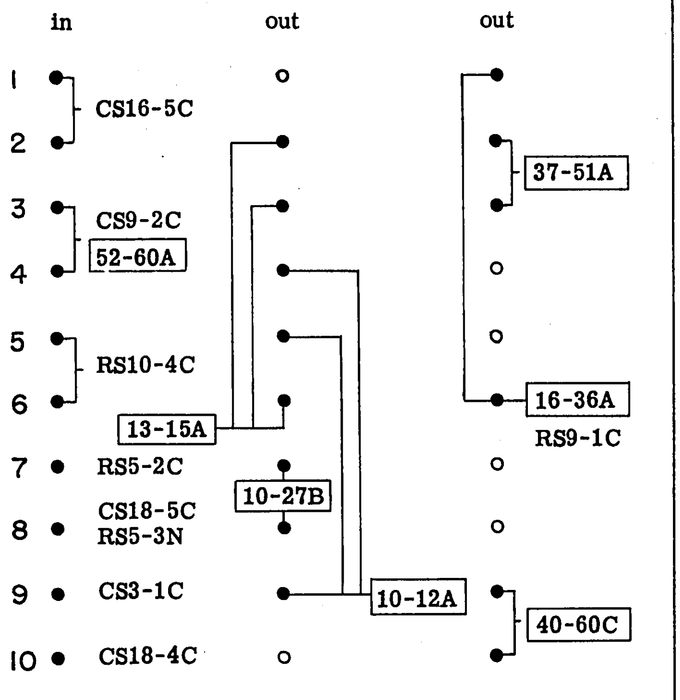
CALCULATE SELECTORS

10	⊕	● Prog Rpt RptDelPU	● BTSS	● RS#5PU	● MQRO	● 1/2 add
		●	●	●	● GS4RO	● MQRI
		● CS4-5N	● CS3-2N	● CS8-5N	● $17B_1$	● $17B_2$
11	⊕	● ctr +	● in 3	● ctr +	● in 5	● in 5
		● MQRI	●	● ×	●	● in 3
		● $28C_2$	● $28C_3$	● $36C_2$	● $36C_3$	● $37C_3$
12	⊕	● M	● emit 8	● ctr-, BTSS	●	●
		●	● GS1&2RO	● ctr +	● in 3	●
		● CS16-3N	● $18B_1$	● $18B_2$	● $18B_3$	●
13	⊕ Ⓣ	● MQRO	● GS3RI	● CS1-3C	● Sup	● RS6-4C, CS3-1T CS16-5T, CS18-5T
		● emit 1	● ctr +	● GS3RI	●	●
		● $8A_1$	● CS16-1N	● CS6-4N	● SF#1	● RS5-2N
14	Ⓣ	● BTSPU#1	● RO	●	●	● Sup
		● in 2	● R&R	● out 2	●	●
		● $10B_3$	● $12B_1$, $15B_1$	● $12B_3$	●	● ⑱, SF#5
15	Ⓢ	● GS4RO	● in 6	● Sup	●	●
		● MQRO	● CS4-5C	●	●	●
		● $3A_1$	● $3A_3$	● ②, SF#1 SF#2	●	●
16	⨸	● ctr -	●	● P	● Sup	● RS6-4C, CS3-1T CS13-5T, CS18-5T
		● CS13-2C	● Sup	● CS12-1C	●	●
		● $8A_2$	● ①	● SF#2	● ⑧	● PSF#1 PSF#2
17	⊗	● ctr -	● ctr +	● MQRI	● ×	● out 6
		● CS7-3C	● CS8-3C	● GS4RI	● FS1&2RI	●
		● $5A_2$	● $7A_2$	● $12B_2$	● $14B_2$	● $16B_3$
18	⊗	● Sup	●	●	● Prog Source	● RS6-4C, CS3-1T CS13-5T, CS16-5T
		● M	●	●	● RS6-3C	●
		● ⑩	●	●	● PSF#10	● RSF#8 RS5-3N

REPEAT SELECTORS

1	$\textcircled{8}$	● M	● P	●	● Prog Source	●
		●	●	● $\boxed{55-60B}$ CS6-5C	●	● Prog Source
		● RS2-1N	● RS2-2N	● RS2-3N	● RS2-4N	● RS2-5N
2	$\textcircled{8}$	● P	● M	●	● Prog Source	●
		● RS1-1C	● RS1-2C	● RS1-3C	● RS1-4C	● RS1-5C
		● SF#8	● SF#10	● RS10-5N	● RS3-4N RS10-4N	● RS3-5N
3	$\textcircled{8}$	● GS3RO	●	●	● Prog Source	●
		● emit 1	● in 6	● Reset	● RS2-4C RS10-4N	● RS2-5C
		● $49B_1$	● $49B_3$	● $50B_3$	● $\boxed{52-54B}$	● RS5-5N
4	$\textcircled{8}$	● Prog Source	● ctr +	● in 6	● ctr +	● in 2
		●	● ctr-, BTSS	●	● GS1&2RI	● in 4
		● $\boxed{1-9, 28-45B}$	● $56B_2$	● $56B_3$	● $51B_2$	● $51B_3$
5	$\textcircled{+}$	●	●	● Prog Source	● Prog Source	●
		● Sup	● CS13-5C	● CS18-5C PSF#8	●	● RS3-5C
		● $\textcircled{13}$	● PSF#7	● $\boxed{28-39C}$	● RS6-3N $\boxed{1-27C}$	● RS6-5N
6	$\textcircled{+}$ $\textcircled{\times}$ $\textcircled{\div}$ $\textcircled{E_x}$	● RptDelDO	● Sup	● Prog Source	●	●
		● RptDelPU	● P	● RS5-4C $\boxed{1-27C}$	● Prog Source	● RS5-5C
		● $57C_2$	● $\textcircled{14}$	● CS18-4N	● CS3-1T, CS13-5T CS16-5T, CS18-5T	● RS7-5N
7	$\textcircled{\surd}$	● RptDelDO	● Sup	●	●	●
		● RptDelPU	●	●	●	● RS6-5C
		● $41A_3$	● $\textcircled{5}$	●	●	● RS8-5N
8	$\textcircled{\surd}$	● FS1&2RO	● ctr +	● Sup	●	●
		● FS3&4RO	● FS3&4RI	●	● Sup	● RS7-5C
		● $54A_1$	● $59A_2$	● $\textcircled{7}$	● $\textcircled{9}$	● $\boxed{1-9A}$
9	$\textcircled{\surd}$	● Prog Source	●	●	●	●
		●	● Prog Rpt	● M	●	●
		● $\boxed{16-36A}$ RSF#1, PSF#6	● $53A_3$	● $\textcircled{19}$	●	●
10	$\textcircled{8}$	● Sup	● $\textcircled{GS4}$	● FS1&2RO	●	● Prog Source
		●	● P	● emit 1	● RS2-4C RS3-4N	● RS2-3C
		● $\textcircled{12}$, PSF#6 PSF#7, PSF#8	● $\textcircled{24}$	● $56B_1$	● PSF#5 PSF#6	● $\boxed{46-51B}$

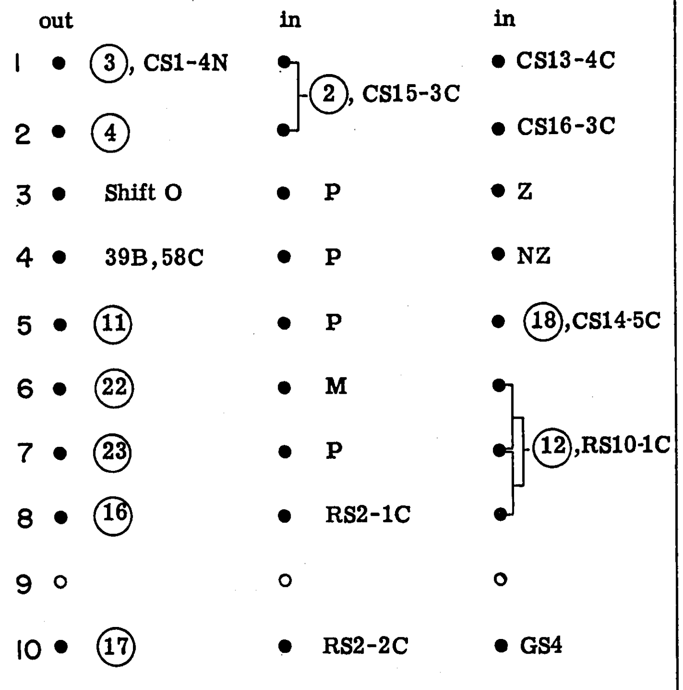
PROGRAM SOURCE FILTERS



PROGRAM EXPANSION

	in	out	out
1	○	● BTSS	● Reset
2	○	● BTSS	● ctr-
3	○	● Prog Rpt	● RptDelPU
4	○	● in 3	● Reset
5	○	○	○
6	○	○	○
7	○	○	○
8	○	○	○
9	○	○	○
10	○	○	○

SUPPRESSION FILTERS



CHANNEL SHIFT

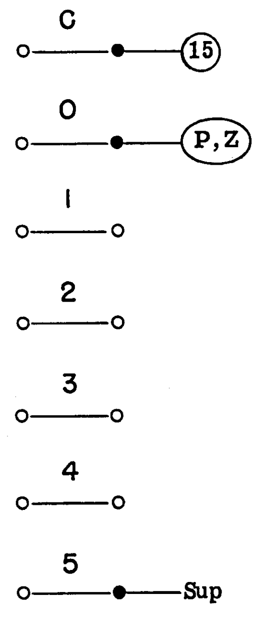


TABLE OF OPERATION WITH ASSOCIATED PROGRAM ACTIVATION, CALCULATE SELECTORS AND REPEAT SELECTORS

Oper. Code	Abbr.	Calculate Selectors Transferred	1st Sweep	2nd Sweep	3 or more Sweeps	Last Sweeps
blank	Ⓣ	13, 14	1-9A, 10-27B	none	----	----
1	⊕	10, 11, 12, 13	1-9A, 10-27B	RS5T 1-27C, 28-39C, 40-60C	RS6T 40-60C	RS6T 40-60C
2	(NR)	5	none	----	----	----
3	⊗	17, 18	1-9A, 10-27B, 28-39C, 40-60C	RS6T 40-60C	RS6T 40-60C	RS6T 40-60C
4	⨸	16	1-9A, 13-15A, 16-36A, 37-51A	RS6T 40-60C	RS6T 40-60C	RS6T 40-60C
5	(√)	7, 8, 9	1-9A, 10-12A, 13-15A, 37-51A, 52-60A	RS7T 10-12A, 13-15A, 37-51A, 52-60A	RS7T 10-12A, 13-15A, 37-51A, 52-60A	RS8 &9T 10-12A, 13-15A, 16-36A, 37-51A, 52-60A
6	Ⓢ	15	1-9A	none	----	----
8	⑧	5, 6	1-9A, 46-51B, 55-60B	RS1T, RS2 or 3 T 10-12A, 13-15A, 16-36A, 52-54B, 55-60B	RS1T, RS2 or 3 T 10-12A, 13-15A, 16-36A, 52-54B, 55-60B	RS4T 1-9B, 28-45B, 46-51B, 52-54B, 65-68B 13 or more Repeats
9	(E_x)	2, 3, 4	1-9A, 10-12A, 40-60C	RS6T 40-60C	none	----
X	Ⓐ	1	1-9A, 10-27B	none	----	----

SUPPRESSION TYPES

P	Suppress on plus balance
M	Suppress on minus balance
Z	Suppress on zero
NZ	Suppress on non-zero
GS1	Group Suppress #1
GS2	Group Suppress #2
GS3	Group Suppress #3
GS4	Group Suppress #4
Sup	Suppress without test

①	Suppress except $\textcircled{\div}$
②	Suppress $\textcircled{S}$
③	Suppress $\textcircled{S}$, $\textcircled{+}$, $\textcircled{T}$
④	P $\textcircled{\div}$; M $\textcircled{+}$; Suppress $\textcircled{S}$
⑤	Suppress $\textcircled{\text{RS7T}}$
⑥	Suppress $\textcircled{8}$, $\textcircled{\text{NR}}$
⑦	Suppress $\textcircled{\text{RS8T}}$
⑧	Suppress $\textcircled{\div}$
⑨	Suppress except $\textcircled{\text{RS8T}}$
⑩	M; Suppress $\textcircled{\times}$
⑪	P; Suppress $\textcircled{T}$
⑫	Suppress $\textcircled{\text{RS10T}}$
⑬	Suppress except $\textcircled{\text{RS5T}}$
⑭	P; Suppress $\textcircled{\text{RS6T}}$
⑮	P, Z, Suppress on operation code 7 [Channel Shift]
⑯	P $\textcircled{\text{RS1T}}$; M $\textcircled{\text{RS2T}}$; Suppress $\textcircled{\text{RS 10T}}$
⑰	M $\textcircled{\text{RS1T}}$; P $\textcircled{\text{RS2T}}$; GS4
⑱	Suppress $\textcircled{T}$
⑲	M $\textcircled{\text{RS8N}}$
⑳	Suppress $\textcircled{E_x}$
㉑	Suppress $\textcircled{S}$, $\textcircled{+}$, $\textcircled{T}$; Do not Suppress $\textcircled{A}$
㉒	M, Suppress $\textcircled{\text{RS10T}}$
㉓	P, Suppress $\textcircled{\text{RS10T}}$
㉔	P; GS4 $\textcircled{\text{RS10T}}$

AN ITERATIVE METHOD FOR SOLUTION OF REGRESSION WEIGHTS AND SIMILAR PROBLEMS

Martin H. Greenberger
Joe H. Ward, Jr.

Human Resources Research Center
Lackland Air Force Base
San Antonio, Texas

I Introduction

The Personnel Research Laboratory concerns itself principally with psychological studies in the scientific placement of airmen in career fields commensurate with their abilities as determined by an aptitude test battery. A frequent project requested of our division, the Machine Processing Branch of Research Services, is for a regression (or beta) weight solution of a matrix of test intercorrelations and a vector of validity (or criterion) coefficients. Stated in more general terms, the problem is to solve a set of n linear, non-homogeneous equations in n unknowns, where the n^{th} order matrix of coefficients is symmetric, has l's down its diagonal, and has all its non-diagonal elements between 1 and -1. We had been attacking the problem by the straight reduction method but were dissatisfied in that this was highly time consuming. It furthermore involved the associating of matrix rows and columns which meant interspersed master card gang punching on the sorter and reproducer. This resulted in still more time delay and placed a maximum of responsibility on the operator's shoulders, providing for a most insecure atmosphere. The results were, moreover, of questionable accuracy. Given only three digits in the original matrix, the numerous multiplications and divisions necessary in the direct solution often produced an accumulated round-off error sufficient to virtually destroy the significance of the result. We could, of course, have circumvented this latter difficulty by assuming five or six digit elements in the correlation matrix; but the necessary additions would have admittedly been too cumbersome for the same control panel and would have demanded a new and still slower control panel.

*The opinions or conclusions contained in this report are those of the authors. They are not to be construed as reflecting the views or indorsement of the Department of the Air Force.

It was apparent that some self-correcting constantly converging iterative technique would be highly desirable. At first we tried the Gauss-Seidel method* which indeed offered the expected improvements, to some degree, but was still disappointing with regard to speed of convergence. The main difficulty of the Gauss-Seidel method seems to be the arbitrariness with which it corrects. It proceeds routinely from equation 1 to equation n, back to equation 1, and so on, without regard to which weight is in greatest error. It was just this consideration that led us to the "educated correction" iterative method with which this paper is concerned.

This method has already had several months of extremely successful operation at our installation. It was used for the first time on the 20th order regression weight problem presented in Section VII. We had previously obtained a three place solution by the Gauss-Seidel method in about 350 iterations at about a minute an iteration. The new "educated correction" procedure gave similar accuracy within 98 iterations, at the same machine time per iteration, representing a time saving of well over 70%. Both methods took zeros as the original approximations to the beta weights. We could, incidentally, have settled for two places available at about the fiftieth iteration. As we later substantiated on other matrices, this meant that a well-behaved system of twenty linear simultaneous equations could now be solved on the 602-A in less than an hour of machine time, by this technique.

Although its speed of convergence relative to other procedures such as the Gauss-Seidel make it most attractive, it should not be supposed that the "educated correction" method has no other advantages for use on the 602-A. It has indeed many further selling features which will occur to the reader during the following development. It also has far wider applicability than has been suggested. Symmetry and the other properties of a correlation matrix are convenient, but certainly not essential for the success of this method. We have been able to extend its use to some matrix inversion, and to many common problems arising in statistical and factor analysis. It is our hope that its possibilities may be recognized in many other fields of study. That is our primary purpose in writing this paper.

*An article on the use of the CPC for the Gauss-Seidel method can be found in the September 1950 issue of IBM's Industrial Computation Seminar Proceedings. The Kelley-Salisbury method, expounded in the Journal of the American Statistical Association, 21:282-292 (1926), more closely resembles the method of this paper in that it, too, suggests correcting on the variable in greatest error. It, however, calculates the optimal correction at each stage, a procedure too involved to be fruitfully incorporated on the 602-A.

II Table of Notation

$r_{ij} = r_{ji}$ .... jth correlation coefficient of ith equation = ith cor. coef. of jth equation

B_i	ith beta weight	prc	proper r column number for a given run
V_i	ith validity coef.	prd	proper r deck number for a given run
e_i	ith error term	Fk	first k card of a given run
x_i	ith approximated B	Lk	last k card of a given run
k	correction term	E_i card	the $(1+23i)$th e card, $i = 1, 2, 3, \dots$
(iii)k	iiith correction term	PEi	program exit i
(iii)	order in which k came	Po	read cycle
(XDI) pu	(X dig or imm) pick up	Pi	program step i
do	drop out	X_i	X in card col i
Si	selector i	D_i	digit in col i
Csi	coselector i	X_k	sign of Fk
CB	control brushes	X_r	sign of r
RB	reading brushes	X_e	sign of e
col(s)	card column(s)	(X if - or no X if +, over low order position)	
MC	multiplicand	/i/	counter i
MP	multiplier	/i, j/	counters i and j
MPL	multiply	(i)	storage unit i
$\longrightarrow$	into		
$\xrightarrow{\text{Si}}$	thru the transferred side of Si into		
$\xrightarrow[\text{Si}]{}$	thru the normal side of Si into		

III Statement of the Method

Consider the solution of the n simultaneous equations

$$r_{i1}B_1 + r_{i2}B_2 + \dots + r_{in}B_n = V_i \quad (i = 1, \dots n) \text{ for } B_1 \dots B_n,$$

where $r_{ij} = r_{ji}$, $r_{ii} = 1$ and $|r_{ij}| < 1$ for $i \neq j$.
Suppose a set of values, $x_1, \dots x_n$ is substituted in these n equations for the unknowns.
Unless $x_1, \dots x_n$ is identical with $B_1, \dots B_n$,

$$r_{i1}x_1 + r_{i2}x_2 + \dots + r_{in}x_n = a_i \text{ will not } = V_i \text{ for all } i = 1 \text{ to } n.$$

Let $V_i - a_i = e_i$ and let e_m be the e_i of greatest absolute value. The iterative method of this paper derives a new variable x'_m for the mth equation:

$$x'_m = x_m + e_m$$

The method always corrects on the variable corresponding to the largest numerical error $|e_m|$. This process is found to bring the $x_1, \ldots x_n$ as close to the $B_1, \ldots B_n$ as is desired, merely by correcting a sufficient number of times. By substituting x'_m for x_m in the equations, a new set of errors (e'_i) are obtained by the formula:

$$e'_i = e_i - r_{im} e_m$$

since $e'_i = V_i - a'_i = V_i - (a_i + r_{im}e_m) = (V_i - a_i) - r_{im}e_m = e_i - r_{im}e_m$.
If subscripts are dropped for the general case, and if e_m is relabeled k, the formula then becomes:

$$e' = e - kr$$

This is the fundamental equation of the method and is the only important operation taking place. The remainder of the programming merely performs operations such as selecting and punching the largest e value ($e_m = k$) after every run, and identifying the equation from which it comes (i.e. the m) by indicating the card column (prc) in which the highest order digit of each r_{im} is punched.

It is convenient to take $x_i = 0$ ($i = 1, \ldots n$) as the first guess to $B_1, \ldots B_n$. Then the first $e_i = V_i$, and $e_m = k$ is just the largest numerical V_i, i.e. V_m. Notice that $e'_m = e_m - k = e_m - e_m$ always $= 0$, which is therefore a check on whether the board is performing correctly.* The approximation to any B_i can be determined at any stage of the procedure merely by adding all the k's of the ith equation which have been computed to that point. The operator can stop any time these approximations to the B_i's are within his desired accuracy.

*See the Supplement to Section VI for the mechanics of this zero check

IV Setup of the Decks

Card	Cols	Prepunched On	Machine Punched On	Information
k	1	all k cards		X_1
	1	first Fk card	all following k cards	prd
	2-3	first Fk card	all following k cards	prc
	4	first Fk card	all following k cards	X_4
	5-8	first Fk card	all following k cards	k
	50-52	all k cards		k order number
	78-80	all k cards		study number
e	3	all e cards		X_3
	3-4	all e cards		equation (row) number
	9-12	all e cards of first e deck		first $e_i = V_i$
	13-76		e cards	e', 4 cols at a time
	77	all e cards		e deck number
	78	E_i cards		X_{78}
	78-80	all e cards		study number
r	1	all r cards		r deck number
	2	all r cards		X_2
	3-4	all r cards		equation (row) number
	9-77	all r cards		r's of a given row 3 digits each
	78-80	all r cards		study number

Note: 4 digit e's and k's, and 3 digit r's are assumed. The number of digits can, of course, be increased by appropriate changes on the panel and cards.

The study and row numbers, and the identifying X's, are to be punched on the cards as indicated in the table above. The other information shall now be further elaborated upon.

k card

The stack of cards which goes through the machine on any run contains two k cards: one k at the front of the deck (Fk), and the following k at the back of the deck (Lk). On the first run Fk has $e_m = V_m$ punched in cols 5-8, prd in col 1, prc in cols 2-3, and X_4. The prd for the first k card is the number of the r deck which contains r_{1m}. The prc is the first card column in which the r_{1m} values are punched. This is the only k card on which the operator must key punch cols 1-8. The machine punches them on all succeeding k cards. The consecutive numbering of cols 50-52 (the order numbers from 000 up) on all k cards is most conveniently done with a consecutive number deck on the reproducer. At the end of the run the machine will punch on the Lk card: k in cols 5-8, prd in col 1 and prc in cols 2-3. The operator will then know which r cols and which r deck to read from on the next iteration, and will be able to adjust his wires accordingly. The same holds true, of course, for successive runs.

e card

The first round of e's are just the V's, so that the first deck of e cards must be key punched with appropriate V's in cols 9-12. Only 16 additional sets of e's (calculated during the determination of (001)k to (016)k by the machine) can be punched on the first e deck, and similarly for successive e decks. The last set of e's punched in cols 73-76 of a given deck, must be reproduced into cols 9-12 of the new e deck. The deck number is punched in col. 77. Rather than make use of the reproducer, the programmer may wish to arrange his 602-A panel so that cols 9-12 of the new e deck can be punched on blank e cards merged with the old e cards of the operating stack at the proper time. X_{78} must be punched on the E_1 cards to indicate to the machine the arrival of a new r deck, as described in Section VI. It is important to be sure that X_{78} is reproduced on the E_1 cards of a newly punched deck.

r card

If $n \leqq 23$, the r card holds a row of the matrix punched in cols 9-77, three cols per r. If, however, $n > 23$ more than one card must be used for each row of the matrix. This entails punching more than one r deck. Twenty-three r's should be punched on the first deck, twenty-three on the second, and so on, until all the r's have been punched. In a 25th order problem, therefore, each row would be punched on two r cards: the first holding twenty-three r's, and the second holding the last two r's. The deck number goes in col 1. Each r deck should be punched twice, once by matrix rows and once by columns. This permits comparison of the two sets on the reproducer as a check. An error indicates either a mistake in punching or lack of symmetry in the matrix; the latter would necessitate checking the source.

Original correlation cards, with a three digit r punched on each card, are often available. In this case the r decks can be much more conveniently punched by running these source cards through a 602-A whose control board is wired to combine up to 23 r's together on a single r card. Whether the r cards are key punched or machine punched, it is generally advisable to have .999 rather than 1.000 for the diagonal r's. This standardizes the use of three digits for all r's. It only slightly alters the values of the first few correction terms and in no way appreciably affects the final result.

V Running the Problem

The final stack of cards, ready to be run through the machine, is in the following order face up:

1. The first Fk card (000)k is on top.
2. The r cards from the first equation (row) are next. If there are say three r decks, the r card from the first row and first r deck is the second card in the stack, followed by the r card from the first row and second deck, followed by the r card from the first row and third deck.
3. The e card of the first row is next.
4. Two and three above repeated for all other rows (from 2 to n).
5. The Lk card (001)k is last and on bottom.

The machine is first cleared with a blank Fk card. This clear card must have an X_1 but neither an X_2, X_3 or X_4. The rightmost skip stop and the following three sets of wires must be moved after every run of the cards through the machine:

1. The four e reading wires which read out of the cols containing the last calculated e.
2. The three r reading wires which read out of the proper r cols (prc) and the one prd wire which reads r col 1 through the digit selector.
3. The four e punching wires which punch the first four cols of the e card immediately following those containing the last calculated e which is being read.

For the first run e shall be read out of cols 9-12, r out of prd and prc, and e shall be punched in cols 13-16. Upon completion of the first run:

1. The Fk card (000)k is filed.
2. The Lk card becomes the Fk card (001)k. (Alternately, k can be stored in the proper unit (1) after each run. With the appropriate changes in wiring this would effectively eliminate the necessity of Fk cards.)
3. (002)k becomes Lk.
4. The e reading wires, the e punching wires and the rightmost skip stop move 4 cols to the right.
5. The r reading wires move to the new prc.
6. The prd wire moves to that digit of the digit selector which equals the new prd.

If the operator has only one control panel to work with, these operations must all be done while the machine is not running. However, if he has two panels at his disposal he can either:

1. Work two (or more) separate problems in which case the only time delay, outside of machine time, is for moving the skip stop. All

other preparations for the second problem can be accomplished while the first problem is running; or,

2. Use both panels on the same problem which allows him to move the e punching and reading wires on the second panel while the first is still running. In any event, however, the k's (in cols 5-8), and the order number (cols 50-52) can always be recorded while the machine is running. This is achieved by writing down this information for a particular k card on the run during which the immediately following k card is Fk.

The k's are conveniently recorded in the following type table. The sum of the k's in the i^{th} column will be the approximation to B_i.

row number	1	2	3	4	----	23	24	----
prd and prc	109	112	115	118	----	175	209	----
	# / k	# / k	# / k	# / k	----	# / k	# / k	----
	# / k	# / k	# / k	# / k	----	# / k	# / k	----
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

#is the k order number, corresponding to the k in the same cell.

For a "well-behaved" matrix this table will have a property which proves extremely useful in the detection of errors. In any given column the k's will uniformly descend toward zero and, after the first entry, will maintain the same sign. That is, every k will be numerically less than all k's above it and will be of the same sign as all k's above it with the possible exception of the topmost k. By a "well-behaved" matrix is meant one whose off diagonal elements are generally small and positive. Matrices other than these, namely those with elements negative and close to 1, will produce many inconsistencies in the above scheme. This checking criterion should not, therefore, be religiously adhered to unless there is no doubt concerning the "well-behaved" nature of the matrix. The array always provides a pattern nevertheless, and one can justifiably become suspicious if too large an increase or sign reversal is noticed.

In addition to the k sheet the operator will have two piles of k cards for each problem:

1. The Lk pile: the blank ones, which are still to become Lk cards,

face up, in ascending order from top to bottom.

2. The Fk pile: the punched ones, which were former Fk cards, face down, in descending order from top to bottom.

After every run, then, a card moves from the top of the Lk pile to the bottom of the operation stack, the former Lk becomes Fk, and the former Fk goes to the top of the Fk pile, face down.

If, after a particular run, the operator wishes to check back several iterations, he merely plugs the e and r reading wires and the prd wires to the same hubs to which they were plugged at the start of the iteration he is interested in. The e punching wires move over four columns as usual. It should be noticed that the e reading and e punching wires will then be no longer adjacent. If on the next run he wishes to continue his checking operation, he moves the wires to their normal, adjacent positions with the prc and prd wires moving as directed by the new Fk card and the e punching wires again moving four columns to the right. On the other hand, if he at any time prefers to revert back to the run on which he left off, he must again move his e punching and reading wires to non-adjacent positions, in the appropriate manner.

The operator should, of course, always use that order and procedure which best allows him to run his problem most economically and efficiently.

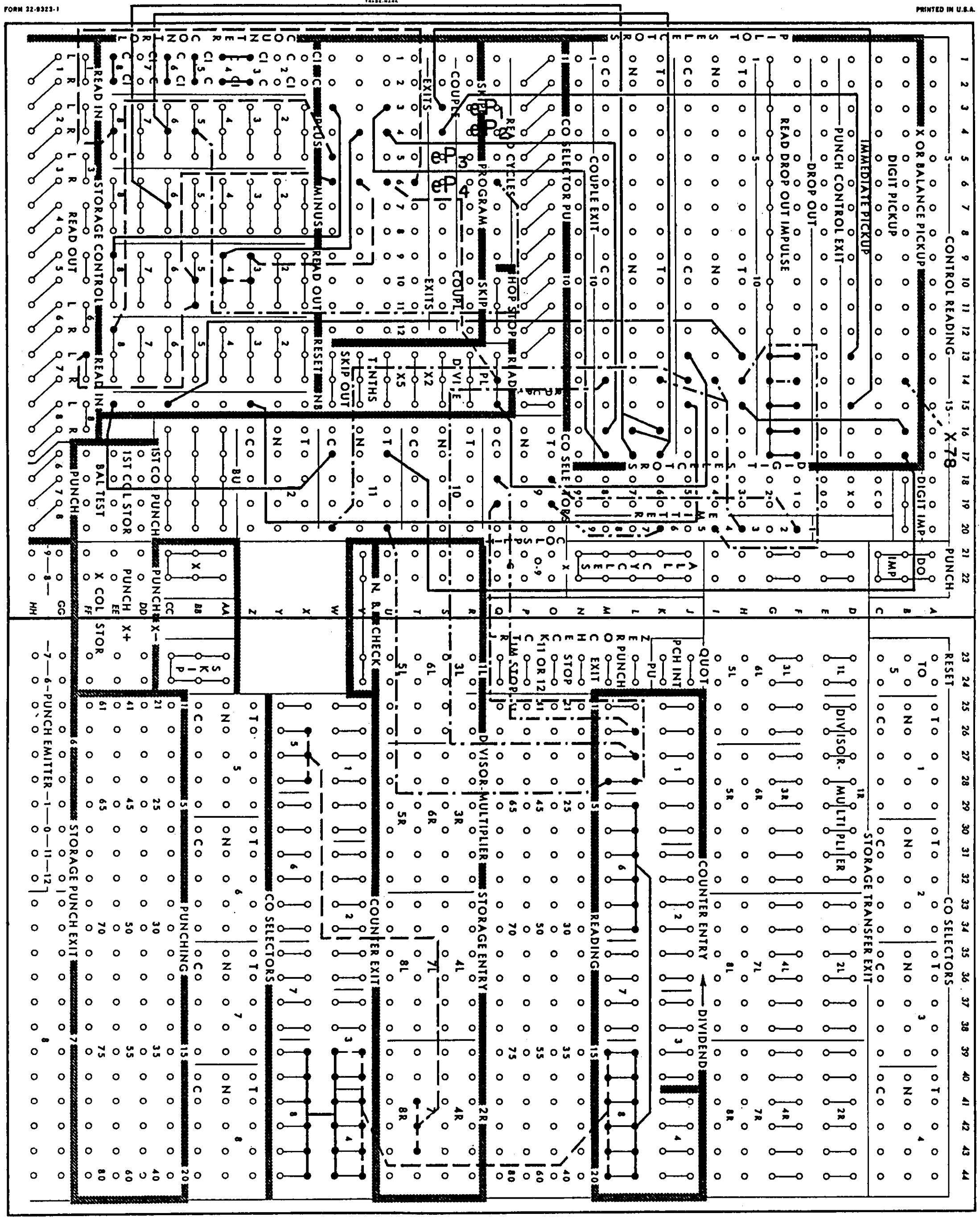
(THIS SIDE MAY BE PUNCHED FOR INSERTION IN NOTEBOOK)

INTERNATIONAL BUSINESS MACHINES CORPORATION

IBM CALCULATING PUNCH
TYPE 602 A CONTROL PANEL

FORM 32-9323-1

PRINTED IN U.S.A.



ELECTRO NO.	CARD NAME OR FUNCTION	X OR D CODE				NOTES: ——wires relating to
	Process whereby prd, prc, and					conditional programming
	em are obtained					—·wires relating to
						prd-prc accumulation

NAME ____________________ DEPT. ____________________

USE ____________________ NO. ____________________

VI Description of Programming

Selector, Coselector Table

S	Cs	Picked up at	How	Dropped out at the end of	Information
1	9	Fk CB	X	Fk RB	X_4 identifying Fk card
2		r CB	X	r RB	X_2 identifying r card
3	11	e CB	X	e RB	X_3 identifying e card
4		Fk and Lk CB	X	Fk and Lk RB	X_1 identifying Fk and Lk cards
5		prd RB	I	prd RB	prd
6		Fk RB	X	Lk RB	X_k
7		r RB	X	e RB	X_r
8		e RB	X	r(or Lk) RB	X_e
10	5,6	start of Lk punching	D	Lk punching	punch control exit of S12
12		Fk and Lk RB	X	following RB	X_1 identifying Lk (and Fk) card
13		e P2	I	e P2	e P2
14		E_i CB	X	E_i RB	X_{78} identifying E_i card
15		e P1	I	e P1	e P1
16		end of e P1	X	r(or Lk) RB	sign of e′ in /3, 4/ at e P1
17		end of e RB	X	r(or Lk) RB	sign of e_m in /8/ at e RB

Note: The following are used interchangeably:
CB.....control brushes.....control time
RB.....reading brushes ....reading time
Pi .....program i..........program i time

Fk Card

CB: X_1 pu S4.
X_4 pu S1.

Po: X_4 to skip out since the Fk card is not to be punched.
|Fk| $\underline{Cs9}\blacktriangleright$(1) storing |Fk| in the multiplier.
X_k pu S6 to be do on Lk card, thus storing the sign of Fk for the entire run.
106 emitted $\underline{Cs9}\blacktriangleright$/5/. The 1 represents r deck one, whereas the 06 provides for an 09, the initial r col, the first time 3 is emitted into /5/.

(PE1) P1: To read.

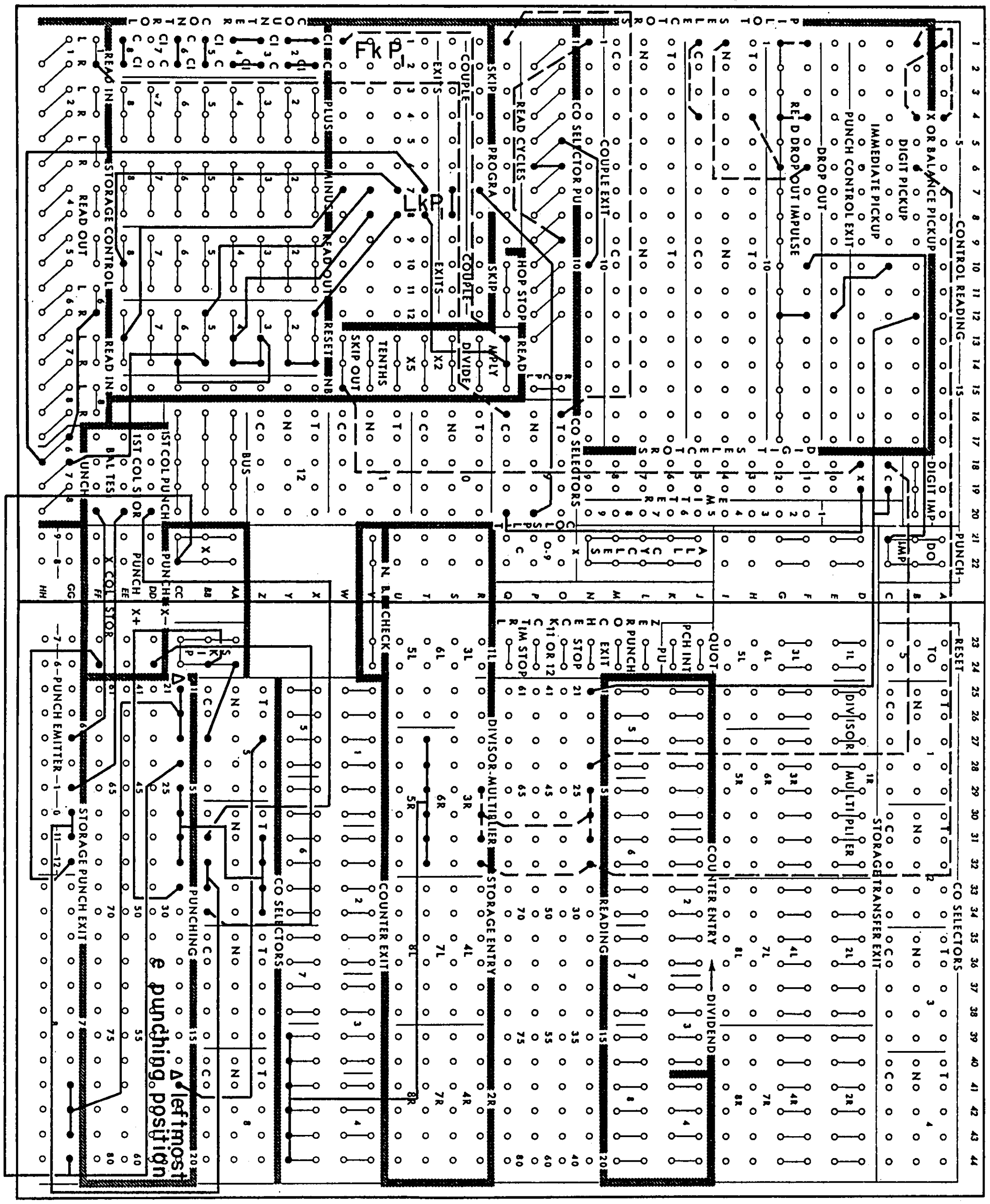
(THIS SIDE MAY BE PUNCHED FOR INSERTION IN NOTEBOOK)

INTERNATIONAL BUSINESS MACHINES CORPORATION

IBM CALCULATING PUNCH
TYPE 602 A CONTROL PANEL

FORM 22-9323-1

PRINTED IN U.S.A.



ELECTRO NO.	CARD NAME OR FUNCTION	X OR D CODE	NOTES:
	Lk and Fk Cards		— — Fk card wires
			—— Lk card wires
			Δ = skip stop

NAME ____________________ DEPT. ____________________

USE ____________________ NO. ____________________

r Card

CB: X_2 pu S2.

Po: X_2 to skip out, skipping out all r decks.
Reset /1,2/, /3,4/ and /6/.
$|r|$ (MC) $\xrightarrow{\quad}$ (2).
X_r pu S7, storing the sign of r.
D_1 goes thru the digit selector $\xrightarrow{S2}$ I pu of S5.
X_2 $\xrightarrow{S5}$ skip to PE2. Improper r decks therefore go to PE1 which causes the next card to be fed and read, whereas the prd skips to PE2 and goes thru the following program step:

(PE2) P1 (2R) MPL $\xrightarrow{\quad}$ /1,2/ giving $|kr|$.
To read.

e Card

CB: X_3 pu S3.
On the E_1 card X_{78} pu S14.

Po: X_3 to skip to PE 3 where the e programs begin.
X_e pu S8, storing the sign of e.

$$\begin{tikzcd}[row sep=small] /1,2/ & S3 \arrow[r, dash] \arrow[dr, dash] & S6 \arrow[r, dash] \arrow[dr, dash] & S7 \arrow[r, equal] & \mp & /3,4/ \text{ giving - kr.} \\ & & S6 \arrow[r, dash] \arrow[ur, dash] & S7 \arrow[r, equal] & \pm & \end{tikzcd}$$

$|e|$ $\xrightarrow{\quad}$ (3).
NB of /8/ $\xrightarrow{Cs11}$ pu S17.
3 emitted $\xrightarrow{Cs11}$ /5/. By adding 3 on each e card we maintain the r col, corresponding to that particular e, in /5/. On the E_1 card, 34 is emitted $\xrightarrow{S14}$ /5/, rather than 3. This effectively changes the 175 to 209 (or 275 to 309 etc), and therefore the first e card of the second deck corresponds to a 209 in /5/, (etc). The two lowest order positions of /5/ will thus indicate the prc and the third lowest order the prd, as the selection process of P3 and P4 makes clearer.

(PE3) P1: (3R) $\xrightarrow[S8\ +]{S8\ -}$ /3,4/ giving e - kr which equals e′.
/8/ $\xrightarrow[S17\ -]{S17\ +}$ /6/ giving - $|e_m|$.
I pu S15.
NB of /3,4/ $\xrightarrow{S15}$ pu S16.

(PE4) P2: I pu S13.
/3,4/ $\xrightarrow{\quad}$ (6) and punch e′ in the four columns immediately following those from which e was just read.
/3,4/ $\xrightarrow[S16\ +]{S16\ -}$ /6/ giving $|e'|$ - $|e_m|$ which, if positive, means there is a new e_m and the next two programs are necessary.

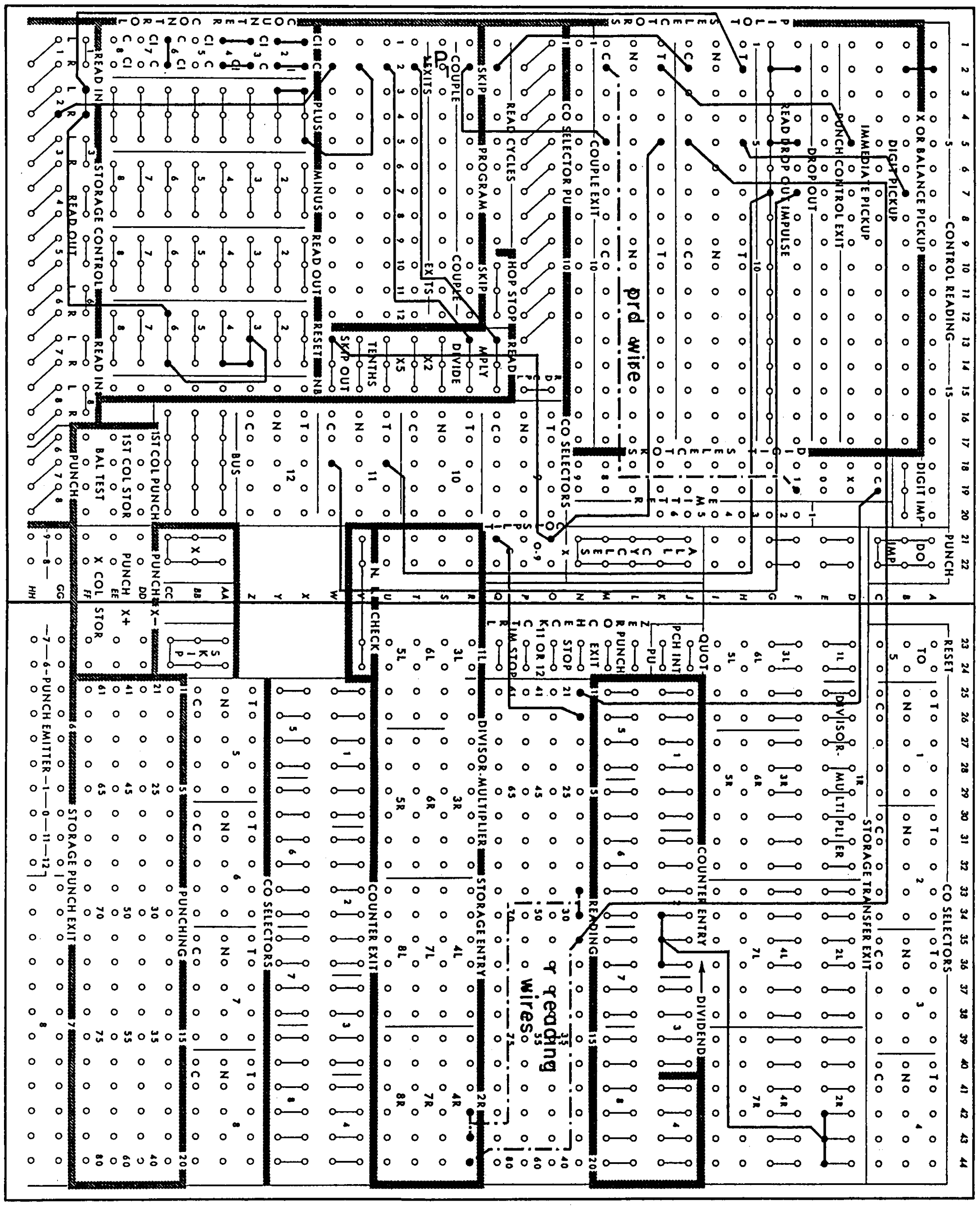
(THIS SIDE MAY BE PUNCHED FOR INSERTION IN NOTEBOOK)

INTERNATIONAL BUSINESS MACHINES CORPORATION

IBM CALCULATING PUNCH
TYPE 602 A CONTROL PANEL

FORM 22-9323-1

PRINTED IN U.S.A.



ELECTRO NO.	CARD NAME OR FUNCTION	X OR D CODE	NOTES:
	r Cards (proper and improper)		
			—· movable wires
			— stationary wires

NAME ______ DEPT. ______

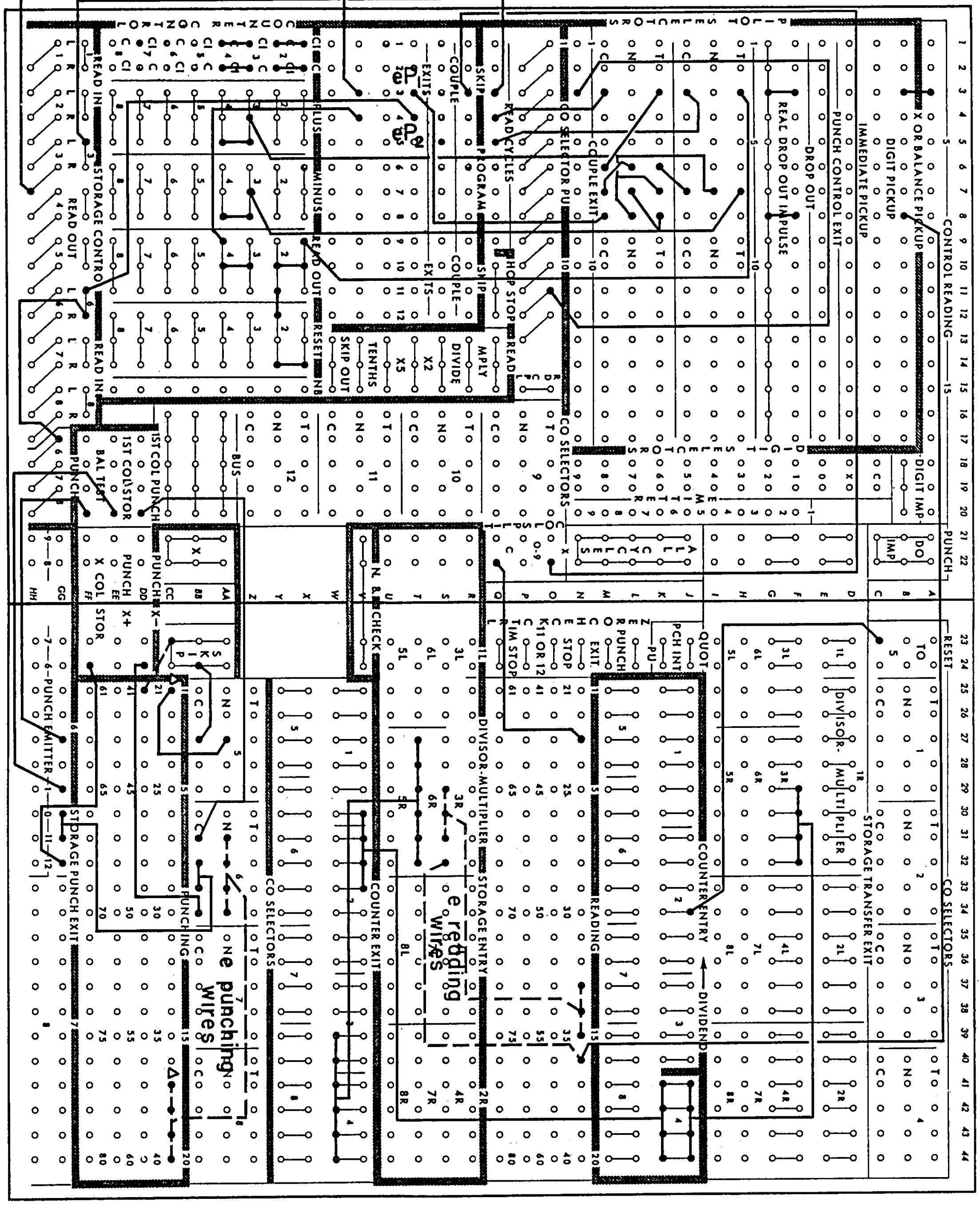
USE ______ NO. ______

(THIS SIDE MAY BE PUNCHED FOR INSERTION IN NOTEBOOK)

INTERNATIONAL BUSINESS MACHINES CORPORATION

IBM CALCULATING PUNCH
TYPE 602 A CONTROL PANEL

FORM 22-9323-1 PRINTED IN U.S.A.



ELECTRO NO.	CARD NAME OR FUNCTION	X OR D CODE				NOTES:
	e Cards					-- movable wires
						— stationary wires
						Δ skip stop

NAME ______________________ DEPT. ______________

USE ______________________ NO. ______________

NB of /6/ $\xrightarrow{S13}$ to 'read' which effectively skips the next two programs if and only if /6/ was negative on P1.

(PE5) P3: (conditional)
Reset /8/. Thus if there is a new e_m, the old e_m has been cleared out to make way for it.

(PE6) P4: (conditional)
/3, 4/$\longrightarrow$/8/ and the new e_m is now in /8/.
/5/$\longrightarrow$(7) clearing (7) and putting in the prc and the prd corresponding to this new e_m. The lowest three positions of /5/ are read into the second, third and fourth positions of (7). An X_4 may then be punched over the (7) units position, which is independent of the other positions of (7) during punching. It is now evident that when (7) is finally impulsed to punch, the prc and prd will represent the r corresponding to the largest of the e_m's; i. e. the e_m of the whole run.
To read.

An interesting question, which could be asked at this point, is whether it is worth the effort to make these last two program steps conditional. The advantage to taking them on each e card regardless of whether or not a sign inversion had occurred, is that the conditional operations could be selected and several non-conditional operations, such as clearing, could be added. However, the fact that the number of sign reversals to be expected with n equations approximates the natural log of n, implied that a substantial amount of time could be saved by ruling out the latter alternative. This observation is derived from simple probability considerations. In order to have a sign reversal on say the i^{th} of n equations $|e'_i|$ must be greater than all $|e'_j|$ where $j = 1, 2, \ldots i - 1$. The probability (and expectation in this case) that this be so for randomly distributed e's is just $1/i$. There is always a sign inversion for e'_1 The probability for this is thus 1 or $1/i$ where $i = 1$. When $i = 2$ the probability is $1/2$, and so on. Now making use of the fact that the expectation of the sum equals the sum of the expectations, the expected number of sign inversions is: $1 + 1/2 + 1/3 + \ldots + 1/n$ which from a study of the area under the curve $y = 1/x$ is seen to be between (ℓn n) and (1 + ℓn n). About three sign inversions might therefore be expected when n = 20. This means that about 34 program steps will, on the average, be saved during each run by making the last two e program steps conditional. This empirically amounted to more than a 15% saving of 602-A time with the 20th order matrix solution listed in Section VII.

<u>Lk Card</u>

CB: X_1 pu S4.
Po: X_1 pu S12.
X_1 $\overrightarrow{Cs9}$ skip to PE7 where the Lk programs begin.

(PE7, 8) P1: Punch (7). This punches the prd in col 1, the prc in cols 2, 3, and both a dummy zero, from the units position of (7), in col 4 and an X_4.
/8/ $\xrightarrow{\quad}$ (6) and punch in cols 5-8.
Reset /1, 2/, /3, 4/, /5/, /6/, and /8/.
The punch control exit of S12 pu S10 when Fk punching starts.
To read.

Supplement (e'_m zero check)

For the incorporation of the zero check the following additions and modifications in setup are necessary:

(1) Rather than punch the e row number in cols 3-4 of the e card as directed in Section IV, the corresponding prd-prc number should be punched in cols 1-3. Thus 109 should be punched instead of 01, 209 instead of 24, 212 instead of 25, etc.

(2) Couple Cs 10 to Cs 9 and Cs 12 to S 13.

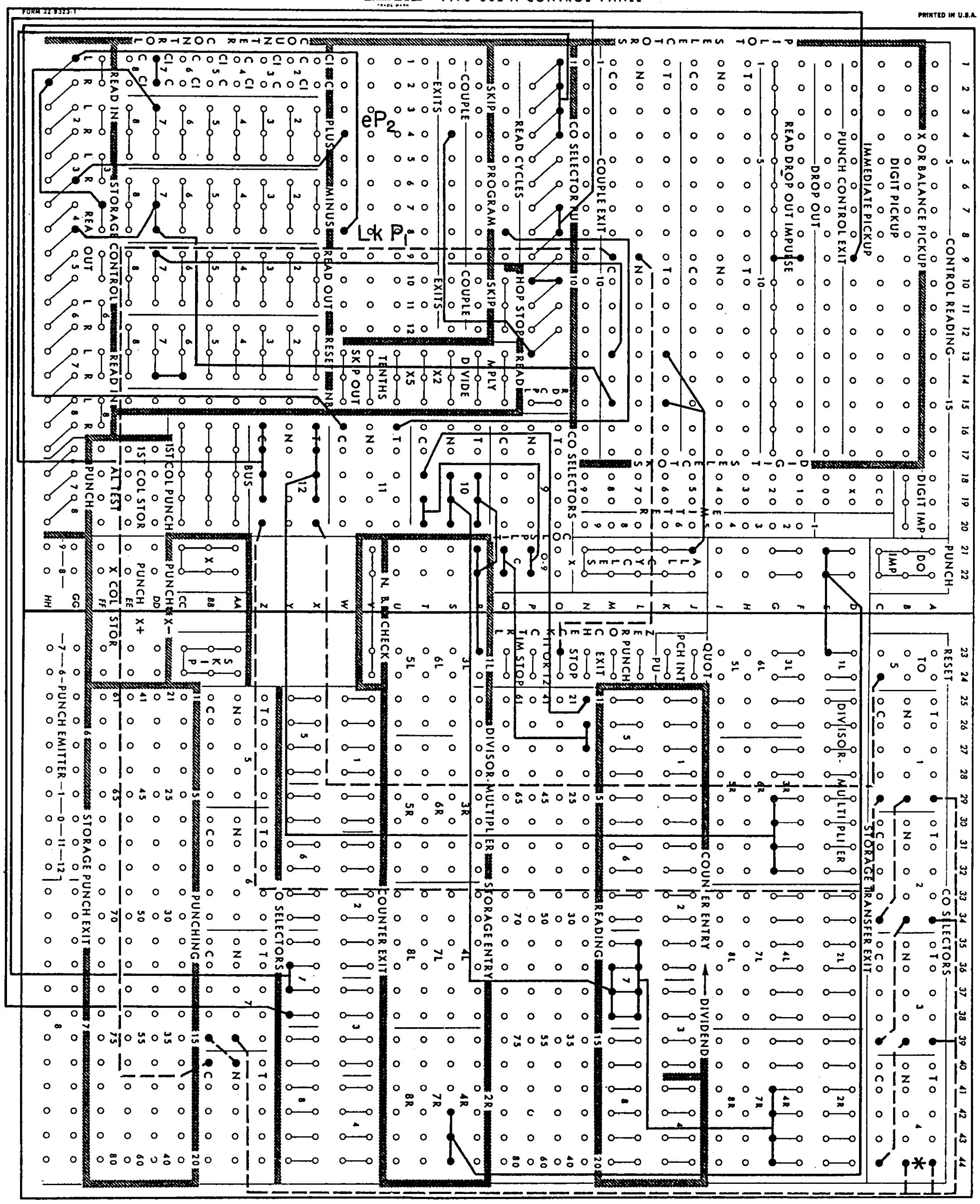
The zero check effectively checks on both operator and machine, from run to run, by making use of the fact that the new error term of the equation just corrected is always equal to zero (see Section III). The board is wired to verify on a given run that the e'_m of the previous run is indeed zero. If it is not, as the corresponding e card goes into the stacker the machine will stop and the compare signal will light up. Since generally none of the original e's, namely the V's, are equal to zero, an error signal on the first run is merely by coincidence and should be ignored. Similar considerations hold for out of sequence runs because of the manner in which the zero check was programmed. This could have been avoided by the more straightforward check of the e'_m of the current run, accomplished by reading the prd-prc number of the run from the Fk card directly into (4R) and by reading the e′ of the run out of /3, 4/. Although this method has the advantage of notifying the operator of a mistake while the pertinent run is still going through the machine, it does not check the transfers of e′ from /3, 4/ to /6/ and to (6), nor does it check the punching of e′. It is therefore not as complete a check as the following:

Fk RB: Fk cols 1-3 $\xrightarrow{\text{Cs 10}}$ (1L) storing the prd-prc of the current run.
r RB : Reset /7/.
e RB : e cols 1-3 $\overrightarrow{\text{Cs 10}}$ /7/ storing the e identification number.
e P1 : (4R) $\xrightarrow{\quad}$ - /7/ subtracting the prd-prc number of the previous run from the identification number of the e card passing through.
e P2 : Rightmost three positions of /7/ $\xrightarrow{\quad}$ pu of Cs 7-8 and Ipu of S9.
Rightmost four positions of (3R) $\xrightarrow{\text{Cs 12}}$ pu of Cs 1-4.
At x-time of this cycle, x (reset to 5) pulses are taken into a C hub of each of the Cs 1-4; the corresponding T hubs are spliced

(THIS SIDE MAY BE PUNCHED FOR INSERTION IN NOTEBOOK)

INTERNATIONAL BUSINESS MACHINES CORPORATION

IBM CALCULATING PUNCH
TYPE 602 A CONTROL PANEL



ELECTRO NO.	CARD NAME OR FUNCTION	X OR D CODE	NOTES:
	e'm Zero Check		
			— — traces path of hot
			impulse [reset to 5 to stop]

NAME *wire to N for first few (e'm)'s which ≠o in lowest order digit; change to and DEPT.

USE keep at T after first compare light — — from then on the (e'm)'s should = o. NO.

together, with the single pulse $\overrightarrow{\text{Cs7 Cs8 S9}}$ Stop. When the prd-prc number of the previous run and the e identification number are one and the same, the difference developed in eP1 will be zero and neither Cs7, Cs8, or S9 will be picked up. If, at the same time, e is different from zero, at least one of Cs 1-4 will be picked up and a pulse will get through to stop the machine. If either e is zero or the difference is non-zero, the pulse will not get through. Since .999's instead of 1.000's are being used for the diagonal matrix entries, the first few (e'_m)'s will not be zero in their lowest order position. To allow for this, the wire leading out of Cs 4 should come from the N hub. After the first compare light, that is when the (e'_m)'s become identically zero, this wire should be changed to the T hub and be kept there until the solution has been obtained.

Lk RB: Reset /7/.

$(1L) \longrightarrow (4R)$ storing the prd-prc of this run for use on the following run.

VII Solution of a 20th Order Regression Weight Problem

Correlation matrix and validity coefficients:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Validities
1.		-123	-049	-466	-222	-240	-073	104	046	-143	-004	004	016	240	017	-050	115	-407	-052	-190	-020
2.	-123		-033	106	106	127	-004	018	063	197	037	037	026	-115	047	005	-012	062	023	023	220
3.	-049	-033		144	046	-024	-032	141	102	-095	-025	254	-085	095	114	028	093	075	-057	-024	048
4.	-466	106	144		341	433	313	045	155	081	116	440	105	083	262	339	019	694	223	403	107
5.	-222	106	046	341		338	325	082	304	208	202	323	287	144	404	358	064	316	536	555	161
6.	-240	127	-024	433	338		208	244	362	408	132	316	287	215	296	274	213	591	413	427	227
7.	-073	-004	-032	313	325	208		072	159	193	321	314	244	087	538	558	-016	305	165	363	115
8.	104	018	141	045	082	244	072		507	083	181	217	256	565	235	352	526	045	332	286	198
9.	046	063	102	155	304	362	159	507		208	236	315	283	616	413	330	520	161	538	455	249
10.	-143	197	-095	081	208	408	193	083	208		109	190	071	-010	110	094	122	197	256	274	143
11.	-004	037	-025	116	202	132	321	181	236	109		183	188	158	357	338	008	102	192	284	039
12.	004	037	254	440	323	316	314	217	315	190	183		197	352	328	381	268	348	396	342	135
13.	016	026	-085	105	287	287	244	256	283	071	188	197		147	470	264	315	175	375	243	128
14.	240	-155	095	083	144	215	087	565	616	-010	158	352	147		234	291	550	-002	480	375	128
15.	017	047	114	262	404	296	538	235	413	110	357	328	470	234		527	176	222	354	323	210
16.	-050	005	028	339	358	274	558	352	330	094	338	381	264	291	527		108	281	350	475	129
17.	115	-012	093	019	064	213	-016	526	520	122	008	268	315	550	176	108		018	355	189	126
18.	-407	062	075	694	316	591	305	045	161	197	102	348	175	-002	222	281	018		233	323	076
19.	-052	023	-057	223	536	413	165	332	538	256	192	396	375	480	354	350	355	233		568	118
20.	-190	023	-024	403	555	427	363	286	455	274	284	342	243	375	323	475	189	323	568		090

Note on matrix: Decimal points are directly in front of the highest order digit; .999's are the diagonal entries.

Correction Table: (2 place numbers have been used for simplicity)

Row Number																			
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
prd and prc																			
109	112	115	118	121	124	127	130	133	136	139	142	145	148	151	154	157	160	163	166
55 0041	01 2043	12 -0008	16 0298	08 0535	02 1110	10 0468	06 0698	00 2490	18 0257	07 -0520	22 0229	41 0082	48 0050	04 0891	33 -0086	21 -0308	11 -0422	09 -0673	03 -0754
94 0008	17 -0229		28 0132	14 0281	12 0408	50 0061	20 0211	05 -0554	37 0078	25 -0193	44 0055	63 0026	76 0019	51 -0043	49 -0056	36 -0074	13 -0241	23 -0202	15 -0290
	35 -0086		39 0093	26 0167	30 0139	69 0022	24 0179	31 -0131	60 0027	98 0005		79 0013	85 0014	68 -0021	58 -0028	47 -0055	19 -0239	43 -0069	27 -0162
	62 -0027		54 0048	34 0098	46 0057	81 0013	32 0098	77 -0016	78 0015					80 -0011	75 -0016	64 -0033	29 -0117	56 -0030	40 -0071
	97 -0006		59 0034	52 0036	65 0023		42 0056							96 -0007	91 -0009	84 -0010	38 -0081	86 -0013	66 -0028
			72 0020	67 0027	87 0009		57 0030										45 -0062		71 -0022
			74 0018	83 0011			70 0020										53 -0037		88 -0009
			89 0008	93 0007			90 0008										61 -0031		
			95 0006														73 -0026		
																	82 -0012		
								Sums after 98th iteration											
0049	1695	-0008	0657	1162	1746	0564	1300	1789	0377	-0708	0284	0121	0083	0809	-0195	-0480	-1268	-0987	-1336
								Guessed corrections											
0006	-0007	0000	0018	0012	0009	0005	0013	-0009	0007	0004	0000	0006	0006	-0005	-0004	-0007	-0019	-0007	-0019
									Final B's										
006	169	-001	068	117	176	057	131	178	038	-070	028	013	009	080	-020	-049	-129	-099	-136

VIII Conclusion

There is, of course, no limit to the size of the matrix, since as many r decks as desired, within reason and practicality, can be punched. As a matter of fact, the larger the matrix the more satisfactory the method, for convergence becomes relatively more rapid with the increase in number of equations. We are currently planning to treat matrices up to the 237th order (item analysis) on the same control panel.

The questions of when to stop the iterations and of how to increase accuracy might be of paramount importance in an organization involved in computational work of a precise nature. We shall, therefore, take them up here even though they are not too significant in our work where two or three places are generally adequate. It should be emphasized that once the error terms dwindle in size to a single digit in their lowest order position, further correction becomes meaningless. Unless additional places (i.e. zeros) are added to the e's and the k's, the correction terms will begin to oscillate around zero without the gaining of any further information. Four digit e's and k's will thus give three place betas at best. Five digit e's and k's would require far fewer iterations for three place beta weights; the additional multiplication cycles necessary, however, might very well cancel out the time saved.

Once the k's begin to dip below 9 in their lowest order, a surprisingly good approximation to the remainder for any beta weight can often be guessed, by observing how the k's for the corresponding B have been behaving. If a variable has been corrected comparatively little, the approximation to the B is probably quite good and the remainder is most likely negligible; e.g., variables 3 and 12 in Section VII. On the other hand, if the variables has been frequently corrected, the remainder is most likely relatively large; e.g., variables 4 and 18. After the iteration process has been terminated, this "guessed" correction can be added at the bottom of each k column to give a final approximation to the B's.

INTERNATIONAL BUSINESS MACHINES CORPORATION
590 Madison Avenue, New York 22, N. Y.

Form No. 34-6191-0-1233P

Litho in USA