IBM®

Technical Newsletter No. 12

APPLIED SCIENCE DIVISION

IBM

APPLIED SCIENCE DIVISION

Technical Newsletter No. 12

June 1956

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

590 Madison Avenue, New York 22, New York

CONTENTS

ROUTINE DERIVATION OF POLYNOMIALS TO REPRESENT FUNCTIONS

Robert W. Schrage
Esso Standard Oil Company
Linden, New Jersey

A problem frequently met in digital computer programming is the suitable representation of given functional relationships. Sometimes an analytic expression for the function is available and no particular programming difficulty exists. Often, however, the function is defined only as a table of corresponding values or as a curve plotted on a chart. When this is the case, there are two methods by which the function can be converted into a form usable on a digital computer. Either a table look-up procedure can be used or an empirical equation may be derived to represent the function. Combinations of both methods can also be used successfully. Most of the single variable type of functions usually met in technical work can be adequately represented by polynomials of a degree no higher than the fourth. Since we have had occasion to use a considerable number of such polynomials in our work, it was desirable to develop a routine method for deriving them. This method has been programmed on the IBM Card-Programmed Electronic Calculator so that only a few minutes are needed to convert a given chart or table into an equation.

The mathematical technique of using orthogonal polynomials is the foundation of the method. Fortunately, it is not necessary to know a great deal of theory to make this application.* For our purposes we have so standardized the calculations that a person totally unfamiliar with either orthogonal polynomials or computers can easily find a polynomial to fit a function. The standardization comes about chiefly through an arbitrary decision to select values of the function at a fixed number of equally spaced argument values, and to use these function values to derive the desired polynomial.

Let us consider the function

$$y = f\ (x). \qquad (1)$$

It is possible to select several function values at equally spaced values of x and derive a polynomial curve which will approximately fit them. This curve may be represented by

$$y = A_0 + A_1g_1 + A_2g_2 + A_3g_3 + A_4g_4 \qquad (2)$$

* See Jack Sherman's paper "The Use of Orthogonal Polynomials in Curve Fitting and Regression Analysis" in the IBM INDUSTRIAL COMPUTATION SEMINAR PROCEEDINGS, September 1950, for a brief, readable introduction to orthogonal polynomials.

where the A's are constants and the g's are first, second, etc., degree polynomials in x.

If these polynomials are of a special type known as orthogonal polynomials, then equation (2) will give a least squares fit of the original function values. This fit will correspond to the highest degree polynomial included in equation (2). The values of the constants, or A's, are independent of the number of terms in equation (2).

Let us assume that we will deal specifically with seven data points. Then the values of the A's up to A_4 may easily be obtained from the following matrix-by-vector multiplication:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -3 & -2 & -1 & 0 & 1 & 2 & 3 \\ 5 & 0 & -3 & -4 & -3 & 0 & 5 \\ -1 & 1 & 1 & 0 & -1 & -1 & 1 \\ 3 & -7 & 1 & 6 & 1 & -7 & 3 \end{bmatrix} \times \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 7\,A_0 \\ 28\,A_1 \\ 84\,A_2 \\ 6\,A_3 \\ 154\,A_4 \end{bmatrix} \qquad (3)$$

Having the A's, the original function is now represented from equation (2) by any degree of polynomial up to and including the fourth.

However it is not convenient to express the g's in terms of x, the original independent variable. Rather than transform x to these several g values every time we wish to use the equation, it would be preferable to transform equation (2) once and for all into a polynomial directly in x.

We can do this in two steps by further matrix-by-vector multiplications. First, we evaluate five constants $a_0, \ldots, a_4$ by the operation

$$\begin{bmatrix} 1 & 0 & -4 & 0 & 6 \\ 0 & 1 & 0 & -7/6 & 0 \\ 0 & 0 & 1 & 0 & -67/12 \\ 0 & 0 & 0 & 1/6 & 0 \\ 0 & 0 & 0 & 0 & 7/12 \end{bmatrix} \times \begin{bmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \\ A_4 \end{bmatrix} = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} \qquad (4)$$

These a values are the coefficients of an equation

$$y = a_0 + a_1 z + a_2 z^2 + a_3 z^3 + a_4 z^4, \qquad (5)$$

where

$$z = \frac{x - x_4}{m} \qquad (6)$$

In equation (6) x_4 is simply the argument value of the fourth or central point in the original seven and m is the constant interval between points. It is frequently convenient to use equation (5) directly because it is easy to evaluate z from equation (6). Furthermore, z has the valuable property that it always lies between ± 3 if equation (5) is used only in the range of the original seven points. This fact is particularly helpful when the polynomial is to be used on a fixed decimal calculator.

If a floating decimal calculator is to be used, however, then it is useful to proceed a step further and calculate the five coefficients $b_0, \ldots, b_4$ from

$$\begin{pmatrix} c_{11} & c_{12} & c_{13} & c_{14} & c_{15} \\ 0 & c_{22} & c_{23} & c_{24} & c_{25} \\ 0 & 0 & c_{33} & c_{34} & c_{35} \\ 0 & 0 & 0 & c_{44} & c_{45} \\ 0 & 0 & 0 & 0 & c_{55} \end{pmatrix} \times \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix} \qquad (7)$$

where

$$\begin{aligned} c_{11} &= 1 \\ c_{12} &= -x_4/m \\ c_{22} &= 1/m \\ c_{13} &= x_4^2/m^2 \\ c_{23} &= -2x_4/m^2 \\ c_{33} &= 1/m^2 \\ c_{14} &= -x_4^3/m^3 \\ c_{24} &= 3x_4^2/m^3 \\ c_{34} &= -3x_4/m^3 \\ c_{44} &= 1/m^3 \\ c_{15} &= x_4^4/m^4 \\ c_{25} &= -4x_4^3/m^4 \\ c_{35} &= 6x_4^2/m^4 \\ c_{45} &= -4x_4/m^4 \\ c_{55} &= 1/m^4 \end{aligned}$$

The b values calculated in equation (7) are the coefficients of a polynomial directly in x:

$$y = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + b_4 x^4 \qquad (8)$$

There may be times when equation (8) can be used successfully on a fixed decimal calculator. This depends, of course, on the scale of x.

It is important to note that although the coefficients $A_0, \ldots, A_4$ are independent of the number of terms in equation (2), this is not true for $a_0, \ldots, a_4$ or $b_0, \ldots, b_4$. These latter coefficients will be different for different degree polynomials. However, values for degrees less than the fourth can easily be calculated by arbitrarily fixing A_4, or A_4 and A_3, etc., as zero.

The matrices in equations (3) and (4) are constant so long as only seven data points are used. If it were desired to use fewer or more points than seven, then new matrices would have to be derived. However, similar techniques would be used. The matrix used in equation (7) does not really depend upon the number of data points.

It is merely the means for transforming a polynomial in the form of equation (5) to that of equation (8). Therefore, provided x_4 and m are generalized to mean the type of constants in equation (6), the matrix in equation (7) does not vary.

Since only small matrix-by-vector multiplications are involved in these calculations, they may be done conveniently on any medium scale calculator. We have developed a program for the CPC using a general purpose floating decimal control panel. This program derives coefficients for all the types and degrees of polynomials discussed in this paper. It also calculates check values so that the fit of the derived polynomials to the original function can be examined. Less than five minutes of calculator time is required for execution of the program.

An outline of the procedure followed in the program is given in the Appendix. A numerical example is also given, showing the various coefficients derived in a typical curve fitting problem. Figure 1 shows graphically the original seven data points used in the problem and a curve representing the fitted fourth degree polynomial. Within the accuracy limitations of the drawing the curve coincides with the original functional relationship.

It may be worthwhile to emphasize the value of calculating check points intermediate between the original data points. Sometimes polynomials may be derived which fit the given data points well enough, but which do not adequately represent the function between them. This difficulty usually arises when the selected points do not adequately describe the curvature of the function. The remedy is either to use a procedure based on more points (and possibly higher degree polynomials) or to divide the function into two or more intervals to be represented by different polynomials.

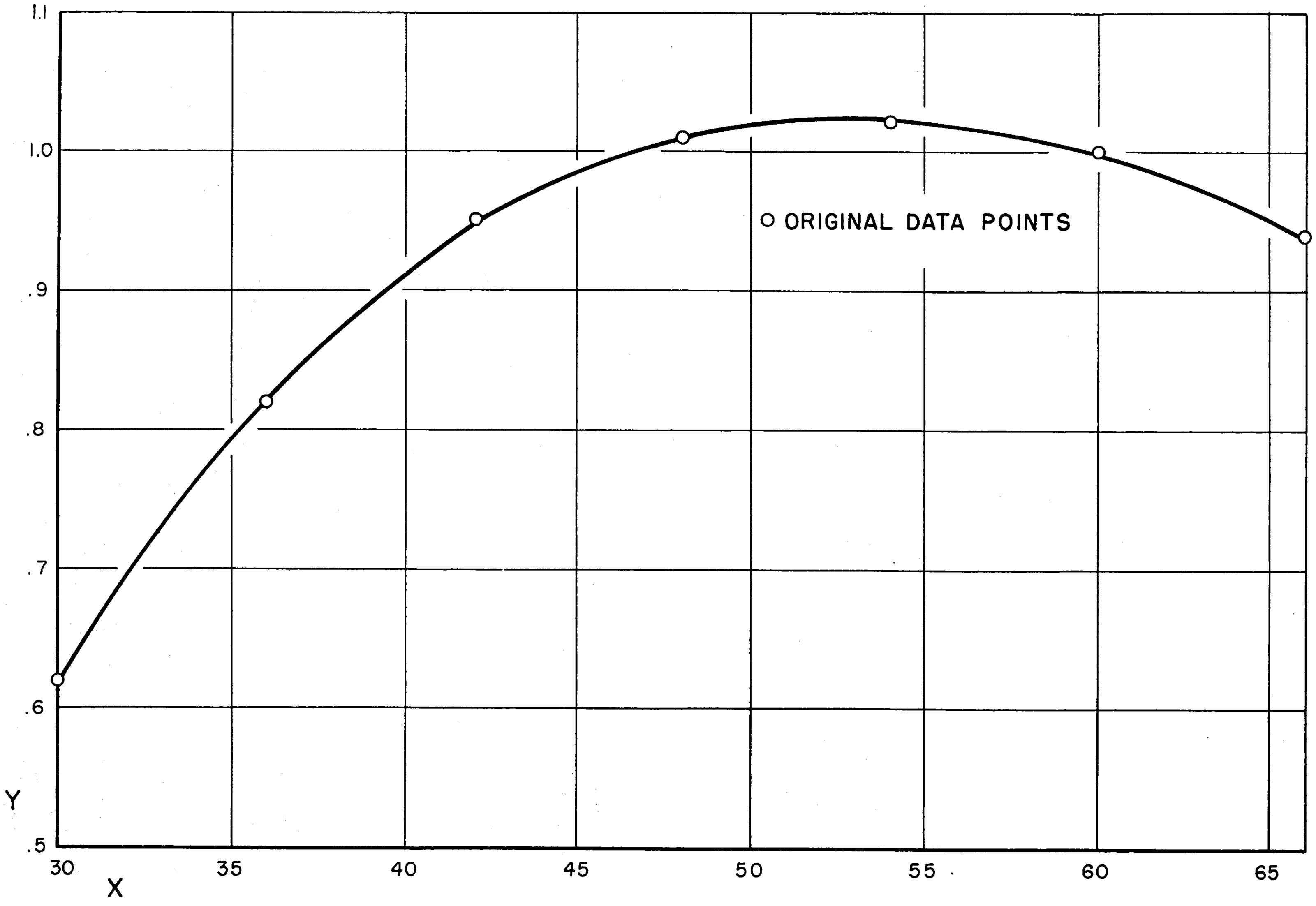


FIG. I FITTING SEVEN DATA POINTS WITH A FOURTH DEGREE POLYNOMIAL

APPENDIX

Program Outline	Numerical Example
1. Reading of problem data. Seven function values, taken at equally spaced values of the argument, are read. The argument value corresponding to the first function value and the increment in argument values are also needed.	$Y_1 = 6.200000 \times 10^{-1}$ $Y_2 = 8.200000 \times 10^{-1}$ $Y_3 = 9.500000 \times 10^{-1}$ $Y_4 = 1.010000 \times 10^{0}$ $Y_5 = 1.020000 \times 10^{0}$ $Y_6 = 1.000000 \times 10^{0}$ $Y_7 = 9.400000 \times 10^{-1}$ $X_1 = 3.000000 \times 10^{1}$ $\Delta X = 6.000000 \times 10^{0}$
2. Calculation of the orthogonal polynomial coefficients used in equation (2). This is done by the matrix-by-vector multiplication given in equation (3).	$A_0 = 9.085714 \times 10^{-1}$ $A_1 = 4.964285 \times 10^{-2}$ $A_2 = -2.559523 \times 10^{-2}$ $A_3 = 1.166666 \times 10^{-2}$ $A_4 = -1.948051 \times 10^{-4}$
3. Calculation of coefficients for the type of polynomial defined in equations (5) and (6). Complete sets of coefficients for all degrees of polynomials up to the fourth are calculated. This is done by the matrix-by-vector multiplication given in equation (4).	1st degree: $a_0 = 9.085714 \times 10^{-1}$ $a_1 = 4.964285 \times 10^{-2}$ 2nd degree: $a_0 = 1.010952 \times 10^{0}$ $a_1 = 4.964285 \times 10^{-2}$ $a_2 = -2.559523 \times 10^{-2}$ 3rd degree: $a_0 = 1.010952 \times 10^{0}$ $a_1 = 3.603175 \times 10^{-2}$ $a_2 = -2.559523 \times 10^{-2}$ $a_3 = 1.944443 \times 10^{-3}$ 4th degree: $a_0 = 1.009784 \times 10^{0}$ $a_1 = 3.603175 \times 10^{-2}$ $a_2 = -2.450757 \times 10^{-2}$ $a_3 = 1.944443 \times 10^{-3}$ $a_4 = -1.136363 \times 10^{-4}$
4. Calculation of coefficients for the final form of polynomial as given in equation (8). The coefficients are obtained by the matrix-by-vector multiplication given in equation (7).	1st degree: $b_0 = 5.114286 \times 10^{-1}$ $b_1 = 8.273805 \times 10^{-3}$ 2nd degree: $b_0 = -1.024284 \times 10^{0}$ $b_1 = 7.652767 \times 10^{-2}$ $b_2 = -7.109779 \times 10^{-4}$ 3rd degree: $b_0 = -1.910950 \times 10^{0}$ $b_1 = 1.364812 \times 10^{-1}$ $b_2 = -2.007271 \times 10^{-3}$ $b_3 = 9.002038 \times 10^{-6}$

Program Outline

5. Calculation of check values. These are obtained by substitution of the appropriate argument values in each of the several degrees of polynomials derived in step 4.* Since these last polynomials depend upon the correctness of all preceding calculations, this is the best place to make the check.

The check values would be the same for any form of polynomial - equations (2), (5), or (8) - of equivalent degree.

Numerical Example

4th degree:

$b_0 = -2.307962 \times 10^{0}$
$b_1 = 1.723686 \times 10^{-1}$
$b_2 = -3.189177 \times 10^{-3}$
$b_3 = 2.583700 \times 10^{-5}$
$b_4 = -8.768216 \times 10^{-8}$

Check values for 4th degree:

$Y_1 = 6.194140 \times 10^{-1}$
$Y_{12} = 7.317100 \times 10^{-1}$
$Y_2 = 8.223130 \times 10^{-1}$
$Y_{23} = 8.934540 \times 10^{-1}$
$Y_3 = 9.471850 \times 10^{-1}$
$Y_{34} = 9.853850 \times 10^{-1}$
$Y_4 = 1.009778 \times 10^{0}$
$Y_{45} = 1.021904 \times 10^{0}$
$Y_5 = 1.023136 \times 10^{0}$
$Y_{56} = 1.014670 \times 10^{0}$
$Y_6 = 9.975520 \times 10^{-1}$
$Y_{67} = 9.726300 \times 10^{-1}$
$Y_7 = 9.405970 \times 10^{-1}$

* To save space, check values for only the fourth degree equations are shown in the numerical example. The intermediate points are at argument values midway between original data points.

AN INFORMATION SEARCHING SYSTEM BASED ON THE IBM COLLATOR

E. R. Lancaster

Department of Mathematics
Newark College of Engineering

From each of a group of documents is selected a set of words which more or less identifies the contents of the document. We prepare a file of these "key" words in such a way that the file can later be searched for an arbitrary set of words and a list of documents obtained, each of which contains all of the words in the set.

The manner in which the file is prepared will be discussed after its operation is described.

Operation of the System

The file consists of IBM punched cards. Following is a column breakdown of the information punched in a typical card prepared for a magazine article. The entries for the first seven columns will be different for other types of documents, such as engineering reports, patent digests, etc.

Columns	Entry
1-3	Source code. 3-letter code for the name of the periodical in which the article appears.
4-5	Last two digits of the year in which the article appeared.
6-7	Number of the month or week in which the article appeared, e.g., 11 for Nov., 31 for July 21, 1954.
8-21	Key word. There is a card for each key word. The placement of words of more than 14 letters in these columns is explained later.
22	Number of card for a given key word. Occasionally a title cannot be fitted into columns 23-80, in which case a second card is necessary. If only one card is required, this column is left blank. Further discussion will be given later.
23-80	Title of article.

The file of cards is arranged in alphabetical order by key word. All cards having the same key word are arranged in numerical order on columns 1-7, 23-24. Cards which are identical in columns 1-21, 23-24 are further arranged in numerical order on column 22.

Suppose we wish to obtain a bibliography on friction clutch materials. First, we refer to a list which gives the number of times each key word appears in the file. If all three of our key words (friction, clutch, material) appear there, we manually remove from the file all cards for these key words. Suppose that of the three decks removed the "friction" deck contains the fewest cards. This deck is placed in the secondary feed hopper of a collator and either of the other decks in the primary feed hopper. Assume we place the "clutch" deck in the primary hopper. The control panel is wired in such a way that the primary deck falls in pocket 2, cards in the secondary deck which match in columns 1-7, 23-24 with cards in the primary deck fall in pocket 3, and the remainder of the secondary deck falls in pocket 4. When this process is completed, we will have in pocket 3 the cards for those articles from which "friction" and "clutch" were originally selected as key words.

The "clutch" deck is taken from pocket 2 and returned to the file. The cards in pocket 4 are placed temporarily in a special rack. The cards in pocket 3 are placed in the secondary hopper, and the "material" deck is placed in the primary hopper. The cards in the "friction" deck which match with cards in the "material" deck again fall in pocket 3, and these are the cards for those articles from which "friction," "clutch," and "material" were originally selected as key words. The "material" deck is taken from pocket 2 and returned to the file. The cards in pocket 4 from the "friction" deck are placed temporarily in the special rack. The information in columns 1-7 and 22-80 of the "friction" deck cards in pocket 3 is listed by means of an IBM accounting machine. If an accounting machine is not available, a list can be made with a typewriter.

This list is the completed bibliography of articles on friction clutch materials. It gives the title of the article, a 3-letter code for the periodical in which it appeared, and the date of publication. Although author's names do not appear in the printed bibliography, they can be used as key words in any search. Author's names appear as key words in the file for every article covered by the file. In the list, the appearance of a 2 just to the left of a title will indicate it is to be read as a continuation of the previously listed title.

Upon completion of the listing, the cards from the "friction" deck which were placed in the special rack are merged, by means of the collator, with the cards used in the listing. The resulting deck (the "friction" deck) is returned to the file.

All the cards for this system may be placed in one file. However, when the number of cards becomes extremely large, it may be more efficient to have several files. This division might be made by classifying the documents covered by the system. A simple example is the division of the cards into mechanical and electrical files. Any such division by subject heading would require the placement of cards for some documents in more than one file. A decision as to which files to search in the preparation of a particular bibliography would be made after examining the lists to determine the number of times each key word appears in the various files.

Preparation of the System

In the following discussion we will continue to consider a magazine article as a typical document.

A person, whom we call the analyzer, selects the key words for a given article. He then writes the results of his analysis on special paper which is ruled into squares in such a way that the squares are easily counted.

The analyzer first rewrites the title of the article with as few letters as possible, using one square of the paper for each letter and space between words. If the revised title has a length greater than 58 squares, letters after the 58th square are placed on the second line.

After the title the analyzer writes on another line the information which will appear in columns 1-7 of the card. He then scans the article for key words, which he underlines and writes on successive lines of the special paper after changing to a basic form those not already in basic form. For example, "experimental" would be changed to "experiment," "verification" to "verify," "properties" to "property." As the work of preparing the file progresses, a dictionary of basic forms is compiled. This dictionary also includes a list of synonyms and other related words to help in the selection of key words for file searches. For example, the synonyms "computer" and "calculator" would be placed in the same word group in the dictionary and only one would ever appear in the file.

Key words exceeding 14 letters in length are decreased to 14 by marking out letters from the right in accordance with the following scheme until the length is 14:

Number of letters	Mark out every
15 to 16	8th
17	5th
18	4th
19 to 21	3rd
22 to 28	2nd

Thus "thermoelectrodynamometer" becomes therolcrdnmmtr."

The analyzer places the completed sheet in one of two stacks, depending on whether one or two lines were required for the title. When the stacks have become sizable, they are given to a key-punch operator.

The key-punch operator first prepares a deck of cards for the stack of sheets with one-line titles. From each sheet are prepared one master card for that sheet and a detail card for every key word on the sheet. On the master card the indicated letters and digits are punched in columns 1-7, an X-punch in column 22, and the title in columns 23-80. The key words are punched, one per detail card, in columns 8-21. The resulting deck is then placed in a reproducer and the information in each master card, with the exception of the X-punch in column 22, is reproduced in the same columns of the detail cards following it. The master cards are removed with a sorter and the remaining cards, called deck A, are stored until work is completed on the sheets with two-line titles.

The key punch operator must go twice through the stack of sheets with two-line titles. The first time through, she punches a master card for each sheet and a detail card for every key word on the sheet. On the master card, columns 1-7 are punched, an X-punch is placed in column 22, and the first line of the title is punched in columns 23-80. Following each master card, a detail card is punched for each key word from the article, as already described for one-line titles. The deck resulting after this first run through the sheets is called deck B. Deck B is now run through a reproducer, and the punches in columns 1-7 and 23-80 of each master card are reproduced in all the detail cards between it and the next master card. Following this, Deck B is again run through the reproducer and a new deck, Deck C, is made which is a duplicate of Deck B in columns 1-22 and blank in columns 23-80. After the master cards are removed with a sorter, Deck B is temporarily stored.

The key punch operator now goes through the sheets with two-line titles a second time, punching one master card but no detail cards for each sheet. These cards have been prepunched with an X-punch and a 2-punch in column 22. The operator punches only the second line of the title on the master card for that title. With a collator these master cards are inserted in Deck C immediately following the master cards already in Deck C. (If the collator is Type 77, the cards should be fed 12's edge first.)

Deck C is now placed in a reproducer, and the punches in columns 23-80 of each master card are reproduced in each detail card between it and the next master card. The master cards are then removed from the deck with a sorter.

Decks A, B, C and the file itself are next sorted into the proper order on the following columns in the order given: 22, 7-1, 21-8.

The single resulting deck is the new file. This deck is placed in an IBM alphanumeric accounting machine, and the words in columns 8-21 are listed without repetition in one column and the number of times the word occurs in another column. If the accounting machine is connected to a reproducer, each word and the number of times it occurs can be punched in cards, one word per card. This facilitates further listing.

If the deck obtained from the reproducer has a color different from the main deck, it can serve another useful purpose. We can, by means of a collator, insert these cards into the main deck in such a way that cards with the same word in columns

8-21 are easily identifiable. This will be helpful in the manual extraction of card groups from the file.

A Useful Auxiliary File

Deck D consists of 792 cards. The first card in this deck has a single hole punched in the 12's row of column 15, the second card has a single punch in the 11's row of column 15, the 13th card has a single punch in the 12's row of column 16, the 32nd card has a single punch in the 5's row of column 17, etc. The 792nd card has a single punch in the 9's row of column 80.

An auxiliary deck is prepared by means of a collator and a reproducer. First, the main file of cards is placed in numerical order on columns 1-7. This deck is divided into 792 parts, each part having approximately the same number of cards. In front of the first part is placed the first card of deck D, in front of the second part is placed the second card of deck D, etc. The resulting deck is called deck E.

Deck F is obtained from deck E in such a way that a given card in deck F has in columns 1-14 a key word from a card of the main file plus the single punch from the card of deck D which precedes the part of deck E containing the card from the main file. Deck F is arranged in alphabetical order on columns 1-14 to obtain deck G. All the cards for a given word in deck G are reproduced on a single card. The resulting deck is the auxiliary deck. It has one and only one card for each key word in the main deck.

The use of the auxiliary deck will be explained by means of an example. Suppose, as in our previous example, we desire a bibliography on friction clutch materials. Befone searching the main file as previously explained, we extract from the auxiliary file the three cards punched respectively with the words "friction," "clutch," and "material." The three cards are placed together in such a way that a punch in columns 15-80 in the same position in all three cards will be apparent. The absence of such a punch is proof that a search of the main file will not reveal an item for the bibliography.

THE STANDARD SCORE METHOD OF COMPUTING AN INTER-CORRELATION MATRIX ON THE 604 ELECTRONIC CALCULATING PUNCH *

K. Warner Schaie
University of Washington

One of the constantly recurring problems in the analysis of social science data is the computation of product-moment correlation matrices. If the correlation data are to be used for further analysis, such as factor analysis or regression, it is necessary to compute the complete matrix, i.e., the correlation of each variable with every other variable. It is clear, then, that when we consider more than four or five variables, manual methods of computation become enormously time-consuming and often prohibitive, even when the number of cases is relatively small.

If we deal with a thousand or more cases, or a hundred or more variables, it will be economical to program and use one of the larger data processing machines such as the 701, 704, 705, or perhaps the 650. In the social sciences, however, the typical problem is not of such a magnitude as to warrant use of these larger machines. Here, in a typical problem we have about 50 to 500 cases, and we are interested perhaps in the correlations of 20 to 50 variables.

Where it is impracticable to use the larger equipment, the use of the 604 Electronic Calculating Punch holds considerable promise, particularly since more than 2,000 of these machines are available throughout the country, their programming is relatively simple, and their cost of operation is fairly reasonable.

This paper describes a method by which it is possible to compute 40th order and larger correlation matrices at an average rate, for a sample of 100 cases, of approximately 100 correlation coefficients per hour, depending upon the skill of the operator. This method will result in the complete computation of the inter-correlation table. Each correlation will be punched to three decimals on a separate card, each of which will also be punched automatically with the correct row and column designation. The check for the computation consists of the independent recalculation of the infra-diagonal values and of the calculation of the diagonal.

THE STANDARD SCORE METHOD

Let us consider the formula for the Pearson Product Moment correlation

* This research was part of an inter-disciplinary study supported by a grant from the Agnes H. Anderson faculty research fund to Dr. Charles R. Strother, Chairman, Committee on Gerontology, University of Washington, Seattle. The writer wishes to acknowledge his indebtedness to the staff of the IBM Seattle sales and service bureaus for their assistance during the machine-testing of the method described in this study.

coefficient to see how it can be written in steps simple enough for efficient machine calculation. To begin,

$$r_{xy} = \frac{\Sigma\, xy}{N\sigma_x \sigma_y}\,, \tag{1}$$

where x is of the form $(X-\overline{X})$ and y is of the form $(Y-\overline{Y})$. Equation (1) indicates that the correlation coefficient between two variables is obtained by summing the cross-products of the deviations of each pair of raw scores from their respective means and dividing the obtained sum by the product of the standard deviations of the two variables times a constant.

It is easily seen that the storage capacity of the calculator (four three-digit units, five five-digit units) as well as the limitations on calculating capacity preclude the solution of this equation in a single operation. A solution well within machine capacity becomes feasible, however, by considering an alternate form of equation (1).

Let z_x be a standard score of the form

$$z_x = \frac{X-\overline{X}}{\sigma_x} \tag{2}$$

and let

$$z_y = \frac{Y-\overline{Y}}{\sigma_y}\ . \tag{3}$$

Then, we can insert (2) and (3) in equation (1) as follows:

$$r_{xy} = \frac{\sum_{1}^{n} z_x\, z_y}{N\sigma_{z_x}\, \sigma_{z_y}} \tag{4}$$

But the standard deviation of any distribution of z-scores equals unity and the denominator term accordingly reduces to the constant N. Furthermore, since the mean of a distribution of z-scores equals zero, all values entering the sub-products of the numerator term will be distributed about zero.

Since the 604 will recognize a negative number (if it is identified by an X-punch over the unit position) and will automatically adjust the sign of products of two negative numbers or of a negative and a positive number, the above development suggests ways of providing sufficient storage space for the simultaneous development of several products over a substantial number of cases, providing that raw scores are first reduced to standard score form.

The number of cases which can be handled by this method is determined by the number of digits to which standard scores are to be computed and, also, by the number of digits to which the sum of products is to be accumulated. The number of correlations which can be computed at one time is limited by the number of storage units available on the calculator. If the maximum of four coefficients at one time is to be computed, standard scores must be expressed in not more than three digits. The maximum number of cases which can be handled can be ascertained by equation (5):

$$N = \frac{10^{b+5}}{10^{2a}} \tag{5}$$

where the superscript a is the number of digits to which standard scores are expressed, the superscript b is the number of decimals to be dropped from each product of two standard scores, and N is an integer, any decimal remainder being dropped. For example, if we plan to compute standard scores to one digit and if we plan to drop no decimals, then a = 1 and b = 0, and inserting these values in (5) gives

$$N = \frac{100000}{100} = 1000.00 \ .$$

Dropping decimals, we would know that in this situation we could handle up to 999 cases.

By full use of punch and calculator selectors available on the 604, it is further possible to compute four sets of four correlations in one machine run. This operation involves the reproduction of cards punched with the standard scores, but makes possible a considerable saving in machine time. In outlining this method we shall use an example of a 40 x 40 correlation matrix for an N of 50 cases (standard scores computed to one decimal), which has actually been computed. A simple modification will also be described by which our method can be expanded to handle up to an 80 x 80 matrix, computed from two basic IBM cards per case. Further modifications are possible, which permit the handling of a 200 x 200 matrix from five basic cards. The maximum utility of our method, however, seems to find an upper limit in an 80th order matrix.

For convenience in presentation, the required procedure will be divided into three parts: (1) conversion of raw scores to standard scores, (2) reproducing the standard score deck and preparing the operating deck, and (3) calculation of correlation coefficients from the operating deck. A further section describes the necessary modifications for calculating an 80 x 80 matrix from two basic cards, while the final section will present a rule of thumb for determining the machine time required to compute a correlation matrix of any given dimension.

1. Conversion of Raw Scores to Deviation Scores

Data Preparation

This method requires that means and standard deviations be known for each variable to be correlated. These values may be either manually computed or obtained in a separate operation on the 604 or other equipment which will accumulate sums of squares.

Raw scores are punched on one or more basic NX cards for each subject. Ideally, raw scores should have no more than two digits. In this case, all 40 scores for a given subject can be punched on one basic card. It is possible to accommodate a few three-digit raw scores (provided the highest digit is identical for all scores) by using an X overpunch. X must never be punched in the unit position, however, as this position is required to identify negative scores.

X-21 cards are punched, one for each variable to be correlated. Columns 1-2 are used to identify the variable; columns 71-75 are assigned to the mean; and columns 76-80 to the standard deviation.

A set of blank cards is punched with a 12-punch in column 61 to receive the deviation scores. One 12-61 card is required for each subject plus two 12-61 cards required for programming operations.

12-61 cards are sorted in with the NX cards so that one 12-61 card follows each NX card. The following card sequence is required for the conversion operation:

12-61	(blank)
X21	(different for each run)
12-61	(blank)
NX	(one pair per subject)
12-61	

Conversion Operation

Figure 1 shows the planning chart for this computation and indicates that an internal check is provided by reverse calculation and zero check before the results are permitted to punch. Figure 2 shows the appropriate wiring diagrams. Wiring condition (1) is a sample of emitting the third digit from an X overpunch.

One run is required for each variable to be correlated. Program advance requires only wiring (2) and (3) to be moved two columns on first reading and punching exits, respectively, after each run.

The same deck of cards is used for each run except that the X21 card is changed after each run to feed in the mean and standard deviation for the variable which is being converted.

Note that we suppress punching on NX cards. If we failed to do this, our basic deck would be destroyed during the operation. Punching is done out of general storage rather than the counter to obviate wiring which would otherwise be required to suppress calculation on the NX cards read cycle.

PROG. NO.	FACTOR STORAGE 1	FACTOR STORAGE 2	FACTOR STORAGE 3	FACTOR STORAGE 4	MULT. QUOT.
READ	RI A 126 (on NX)	RI B 90.25 (on X21)		RI C 12.15 (on X21)	

	GENERAL STORAGE 1	GENERAL STORAGE 2	GENERAL STORAGE 3	GENERAL STORAGE 4
READ				

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

ELECTRONIC CALCULATING PUNCH, TYPE 604

PLANNING CHART

FORM 22-9322-5
PRINTED IN U.S.A.

APPLICATION $X = A,\ \bar{X} = B,\ \sigma = C,\ (X - \bar{X}) = D,\ \frac{x}{\sigma} = E$

PROBLEM $\frac{X - \bar{X}}{\sigma} = \frac{x}{\sigma}\downarrow$

PROG. NO.	NOTES	READ UNITS INTO	READ UNITS OUT OF	PROGRAM SUPPRESS	PROG. NO.	FACTOR STORAGE ASSIGNMENT 1	FACTOR STORAGE ASSIGNMENT 2	FACTOR STORAGE ASSIGNMENT 3	FACTOR STORAGE ASSIGNMENT 4	MULT. QUOT.	COUNTER	GENERAL STORAGE ASSIGNMENT 1	GENERAL STORAGE ASSIGNMENT 2	GENERAL STORAGE ASSIGNMENT 3	GENERAL STORAGE ASSIGNMENT 4	PROG. NO.
					READ	RI A 126 (on NX)	RI B 90.25 (on X21)		RI C 12.15 (on X21)		▨					READ
1	RO A; Ri 5th +	5			1	RO A 126					+ A 126.					
2	RO B; Ri 3rd -	3			2		RO B 90.25				- B 35.75					
3	RO C; Divide				3				RO C 12.15	E 2.94	29					
4	RR				4						RR					
5	RO C; Multiply				5				RO C 12.15		C x E = D 35.7210					
6	RO B; Ri 3rd +	3			6		RO B 90.25				D+ B = 125.9710					
7	RO A; Ri 5th -	5			7	RO A 126					- A 000.029					
8	Zero check in 5th	5			8						↓					
9	RO E; RI +				9					RO E 2.94	2.94					
10	1/2 Adjust				10						5 2.99					
11	RR; RIGS 4; Ro 2nd		2		11						RR				RI E 2.9	
PUNCH				NX & X21	PCH	▨	▨	▨	▨	▨					P 2.9	PCH

PUNCH					
COUNTER					
GENERAL STORAGE					P 2.9

Figure 1. Planning Chart for Conversion of Raw Scores to Standard Scores

INTERNATIONAL BUSINESS MACHINES CORPORATION

ELECTRONIC CALCULATING PUNCH, TYPE 604 CONTROL PANEL

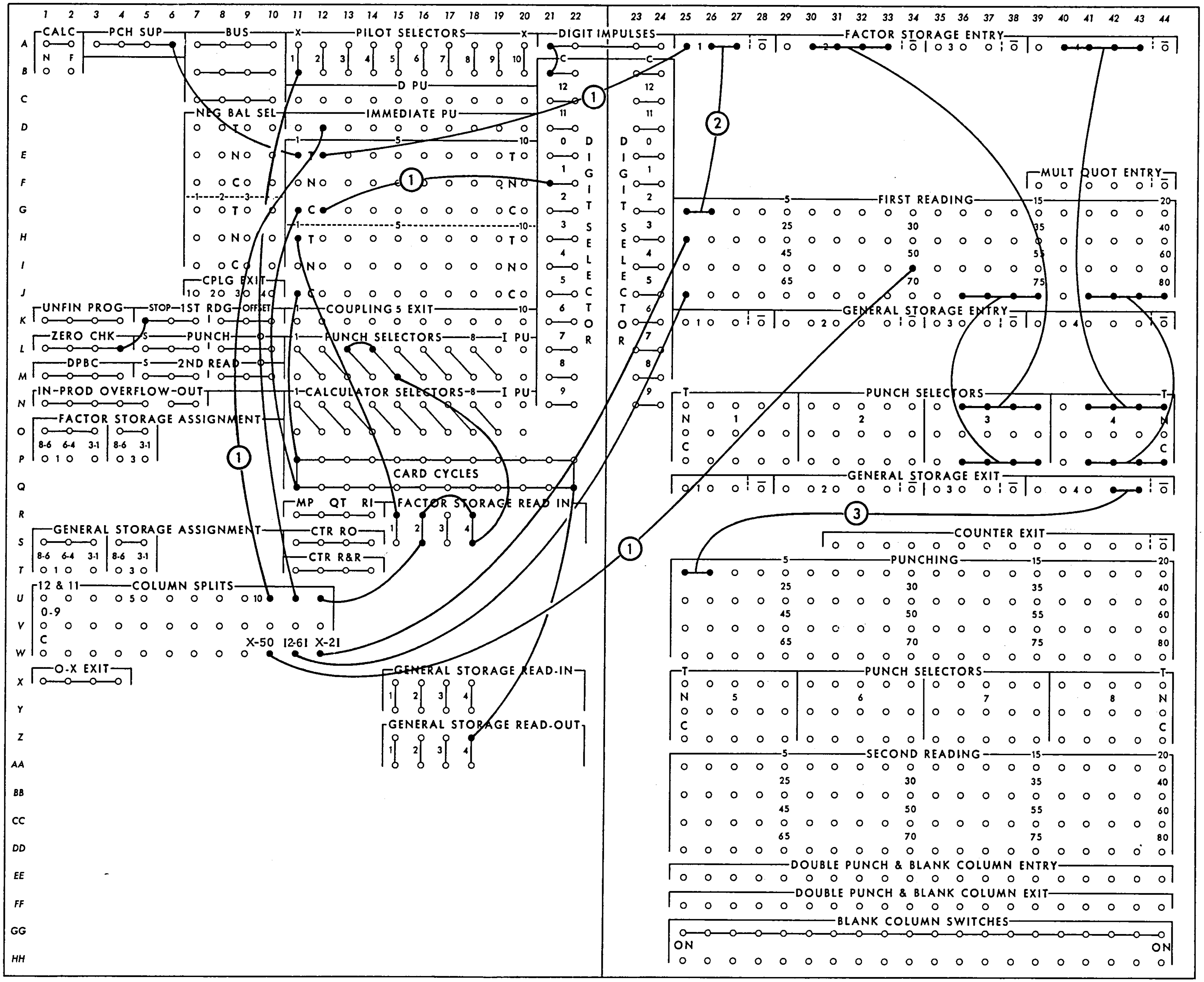


Figure 2 . Wiring Diagram for Converting Raw Scores to Standard Scores

2. Preparing the Operating Deck

In order to minimize program advances, we now reproduce the 12-61 deck three times. The 12-punch on these reproduced decks must be transferred, however, to 12-65, 12-67, and 12-69, respectively. The reason for this change will become apparent later on. Next, we need a set of cards, one for each variable, each having the variable designation punched in columns 1-2 of the card, and an X-punch in columns 61 and 79. For smoothness of operation it might be worthwhile to duplicate the complete set so far described, so that one set can be arranged in sequence while the other is feeding through the calculating punch. Note that 12, rather than X-punches, must be used if negative numbers are to be handled.

Next, we need a set of blank cards to receive the correlation coefficients and their variable identifications. Since we will calculate four correlations at one time, we need four different sets to program the separate conditions. We shall designate these cards X1, X3, X5, and X7, respectively. The number of these cards needed is obtained by dividing the total number of correlations to be computed (both original and reverse calculation) by four. In our example we have 40 x 40 or 1600 correlations, which means that we need 400 each of the X1, X3, X5, and X7 cards.

The card sequence for the final computation will be as follows:

X7	(blank)
12-61	(one for each subject)
X79	(different for each run)
X1	
X3	
X5	
X7	
12-65	(one for each subject)
X79	(different for each run)
X1	
X3	
X5	
X7	
12-67	(one for each subject)
X79	(different for each run)
X1	
X3	
X5	
X7	
12-69	(one for each subject)
X79	(different for each run)
X1	
X3	
X5	
X7	

A new blank set of X1, X3, X5, and X7 cards is used for each run. For a sample of 50 subjects this would imply an operating deck of 221 cards. Of these, only 20 cards would be changed on each run, i.e., we would need to insert new cards to replace the ones on which correlations have been punched and also to insert the appropriate variable designations given by the X79 cards. For convenience in making these changes, the X79 and X1-X7 cards should be punched on distinctive colors.

IBM

INTERNATIONAL BUSINESS MAC
ELECTRONIC CALCULATING
PLANNING CH

APPLICATION ________ PROBLEM

$\sum_{1}^{N}(A\times B)/N = r_{ab}$; ↓ $\sum_{1}^{N}(A\times C)/N = r_{ac}$; ↓

$\sum_{1}^{N}(A\times D)/N = r_{ad}$; ↓ $\sum_{1}^{N}(A\times E)/N = r_{ae}$; ↓

READ	FACTOR STORAGE 1	2	3	4	MULT. QUOT.		GENERAL STORAGE 1	2	3	4		
	RI B		RI C		RI A		RI D		RI E			

Prog. No.	NOTES	Read Units Into / Out of	Program Suppress	Prog. No.	Factor Storage Assignment 1	2	3	4	Mult. Quot.	Counter	General Storage Assignment 1	2	3	4
READ				READ	RI B		RI C		RI A		RI D		RI E	
1	RO B; Multiply			1	RO B					A x B				
2	RO ΣAB; RI+		1st NX	2		RO ΣAB				+ ΣAB				
3	RR; Store ΣAB			3		RI ΣAB				RR				
4	RO C; Multiply			4			RO C			A x C				
5	RO ΣAC; RI+		1st NX	5				RO ΣAC		+ ΣAC				
6	RR; Store ΣAC			6				RI ΣAC		RR				
7	RO D; Multiply			7						A x D	RO D			
8	RO ΣAD; RI+		1st NX	8						+ ΣAD		RO ΣAD		
9	RR; Store ΣAD			9						RR		RI ΣAD		
10	RO E; Multiply			10						A x E			RO E	
11	RO ΣAE; RI+		1st NX	11						+ ΣAE				RO ΣAE
12	RR; Store ΣAE			12						RR				RI ΣAE
13*	RO ΣAB, ΣAC, ΣAD or ΣAE into 2nd	2	NX	13						+ ΣAB, ΣAC, ΣAD or ΣAE				
14	Emit 5; Divide		NX	14					r	Remainder				
15	RR		NX	15						RR				
16	RO r; RI+		NX	16						+ r				
17	1/2 Adjust		NX	17						5				
PUNCH	COUNTER: RR & Punch x xxx GENERAL STORAGE	12-61, 12-67	12-65, 12-69	PCH						RR & Punch xxxx				

*The four sums of products are selectively read into the counter depending on whether calculator selectors 2, 3, 4, or 5 are activated.

Figure 3. Planning Chart for Correlation from Standard Scores.

3. Computing the Correlation Matrix

Figure 3 gives the basic planning chart for this operation. Standard scores are read in through the four three-position storage units and the MQ unit, and the sums of products are accumulated in the four five-position storage units. The capacity of these units limits the size of the sample that can be manipulated in one operation.

The X1-X7 cards are used to select the sum of products to be divided by the sample size, which will result in the correlation coefficient. N (the sample size) is emitted from the digit emitter during division. If this divisor has more than one digit, it will be necessary to add the required number of programs to develop the divisor in the counter and transfer to a free three-position storage unit. If the divisor has more than three digits, it will be necessary to program division in two separate program steps.

Figure 4 gives the wiring diagram for this operation. Note that punching is suppressed on all but X1-X7 cards. This design provides for the computation of 16 correlations in one machine run. This is accomplished by successively computing the correlation between the set of four variables A, B, C, and D and the variables E, F, G, and H, which are read into the MQ unit successively by wiring through punch selectors. This is the reason why we reproduce the original standard score set (12-61) with the classifications 12-65, 12-67, and 12-69, respectively. These 12-punches select the multiplier variable with which the four first variables now in storage are to be multiplied.

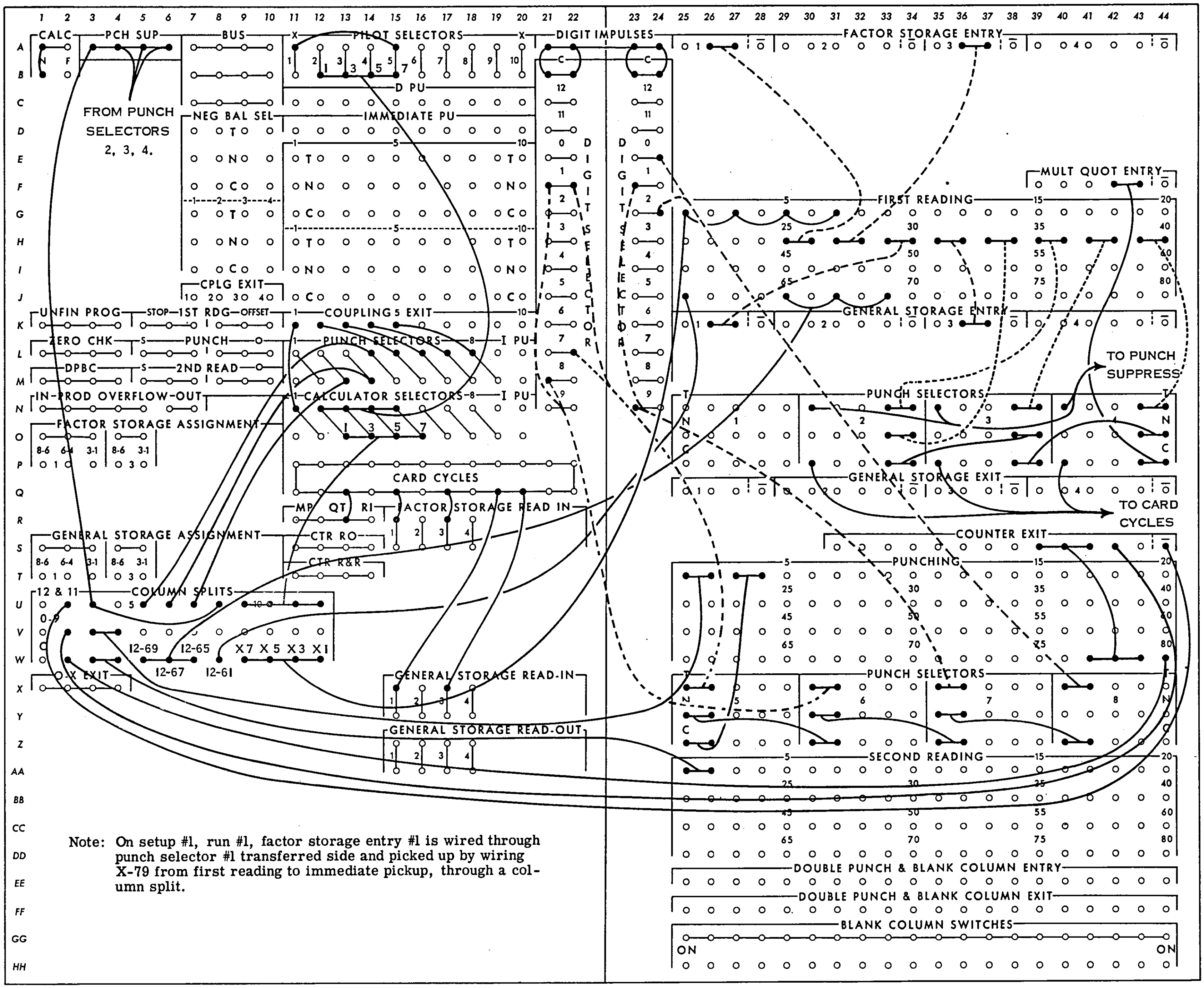
Variable designations are provided by reading in the row designation from the X-79 cards at Second Reading and punching directly from Second Reading on X1, X3, X5, and X7. On the first run we must eliminate reading from columns 1 and 2. On the NX card this is done by punching an X-79 into NX01 and wiring FS1, using punch selector 1 transferred, picked up by X-79. This wiring may be discarded after the first run. Column designations are punched from digit emitters which are selected through punch selectors picked up by X1-X7 cards to punch correct designation.

Program Advance Schedule

In figure 4, solid lines are wiring which will remain throughout the operation; dashed lines indicate wires which must be changed for each set-up; and dotted lines indicate wires which must be changed for each run. In our example, 16 wires must be moved for each set-up, while 8 wires are moved for each run.

The number of set-ups required is determined by dividing the number of variables to be correlated by four. Each set-up again has as many runs as there are set-ups. For the 40 x 40 matrix, then, we need 10 set-ups with 10 runs each, or a total of 100 machine runs of 421 cards. The program advances involve 9 major and 90

ELECTRONIC CALCULATING PUNCH, TYPE 604 CONTROL PANEL



Note: On setup #1, run #1, factor storage entry #1 is wired through punch selector #1 transferred side and picked up by wiring X-79 from first reading to immediate pickup, through a column split.

Figure 4. Wiring Diagram for Computing Correlation from Standard Scores (sample for set-up #4, run #5)

minor changes, each of which, however, should not take more than a minute or two. Figure 5 shows a sample program-advance schedule for one set-up.

MODIFICATION FOR COMPUTING AN 80 x 80 INTERCORRELATION MATRIX

The method described for the 40 x 40 table can be used for the larger matrix except that one additional punch selector must be wired on all set-ups where the multiplicand and multiplier are read from two different cards. It is also necessary to wire the digit selector through a pilot selector to get the correct variable designation. In the 80-variable case, this will involve changing 16 instead of 8 wires for each set-up.

By using the formula described earlier, we find that in the 80-variable case we require 20 set-ups of 20 runs each, or a total of 400 runs. Half these runs will compute the intercorrelations on one card, and thus there will be 221 card runs for a sample of 50 cases. The other half of the operation will involve runs with information read from two basic cards, and thus there will be 421 card runs.

The number of runs can be halved if we introduce some relatively simple wiring modifications. The principle involved in this change is that we will successively compute the correlation between a set of four multiplicands on card 1 and a set of four multipliers, first on card 1 and then on card 2. On the next run we simply reverse this operation and calculate the correlations between a set of four multiplicands on card 2 and sets of four multipliers first on card 2 and then on card 1. The change between the two halves of this operation merely involves transposing the wiring from the normal and transferred exits of the pilot selectors, determining whether card 1 or 2 is to be read.

A special function punch is needed to activate the pilot selectors. It is necessary, therefore, to emit a 12-61 punch into all cards 1, and a 12-63 punch into all cards 2. In setting up the operating deck for the two-card operation, it should be remembered that we need first a deck of the same composition as described in the preceding section, and then another deck obtained by collating decks composed of cards 1 and cards 2, respectively.

With this modification we will require 10 set-ups of 20 runs each, where each run will involve 641 cards and where 32 correlations will be computed in each run. Figure 6 shows the additional wiring required to control the 80-variable condition. Program-advance schedules can easily be designed by the appropriate expansion of figure 5.

PROGRAM ADVANCE SCHEDULE (Correlation from standard scores) IBM 604

SET-UP: # 4		PS 5T to DS 13
FS 1 to RI 25 & 26	GS 1 to RI 29 & 30	PS 6T to DS 14 PS 7T to DS 15
FS 3 to RI 27 & 28	GS 3 to RI 31 & 32	PS 8T to DS 16

1st run PS 2N to col. spl. # 1 & RI 2 PS 2T to " " # 2 & RI 4 PS 3T to " " # 3 & RI 6 PS 4T to " " # 4 & RI 8	6th run PS 2N to RI 41 & 42 PS 2T to RI 43 & 44 PS 3T to RI 45 & 46 PS 4T to RI 47 & 48
2nd run PS 2N to RI 9 & 10 PS 2T to RI 11 & 12 PS 3T to RI 13 & 14 PS 4T to RI 15 & 16	7th run PS 2N to RI 49 & 50 PS 2T to RI 51 & 52 PS 3T to RI 53 & 54 PS 4T to RI 55 & 56
3rd run PS 2N to RI 17 & 18 PS 2T to RI 19 & 20 PS 3T to RI 21 & 22 PS 4T to RI 23 & 24	8th run PS 2N to RI 57 & 58 PS 2T to RI 59 & 60 PS 3T to col spl. # 5 & RI 62 PS 4T to RI 63 & 64
4th run PS 2N to RI 25 & 26 PS 2T to RI 27 & 28 PS 3T to RI 29 & 30 PS 4T to RI 31 & 32	9th run PS 2N to col. spl. # 7 & RI 66 PS 2T to " " # 8 & RI 68 PS 3T to " " # 9 & RI 70 PS 4T to RI 71 & 72
5th run PS 2N to RI 33 & 34 PS 2T to RI 35 & 36 PS 3T to RI 37 & 38 PS 4T to RI 39 & 40	10th run PS 2N to RI 73 & 74 PS 2T to RI 75 & 76 PS 3T to RI 77 & 78 PS 4T to RI 79 & 80

Special Wiring on Run # 4 Split-wire

PS 2N to FS 1 PS 2T to FS 3 PS 3T to GS 1 PS 4T to GS 3

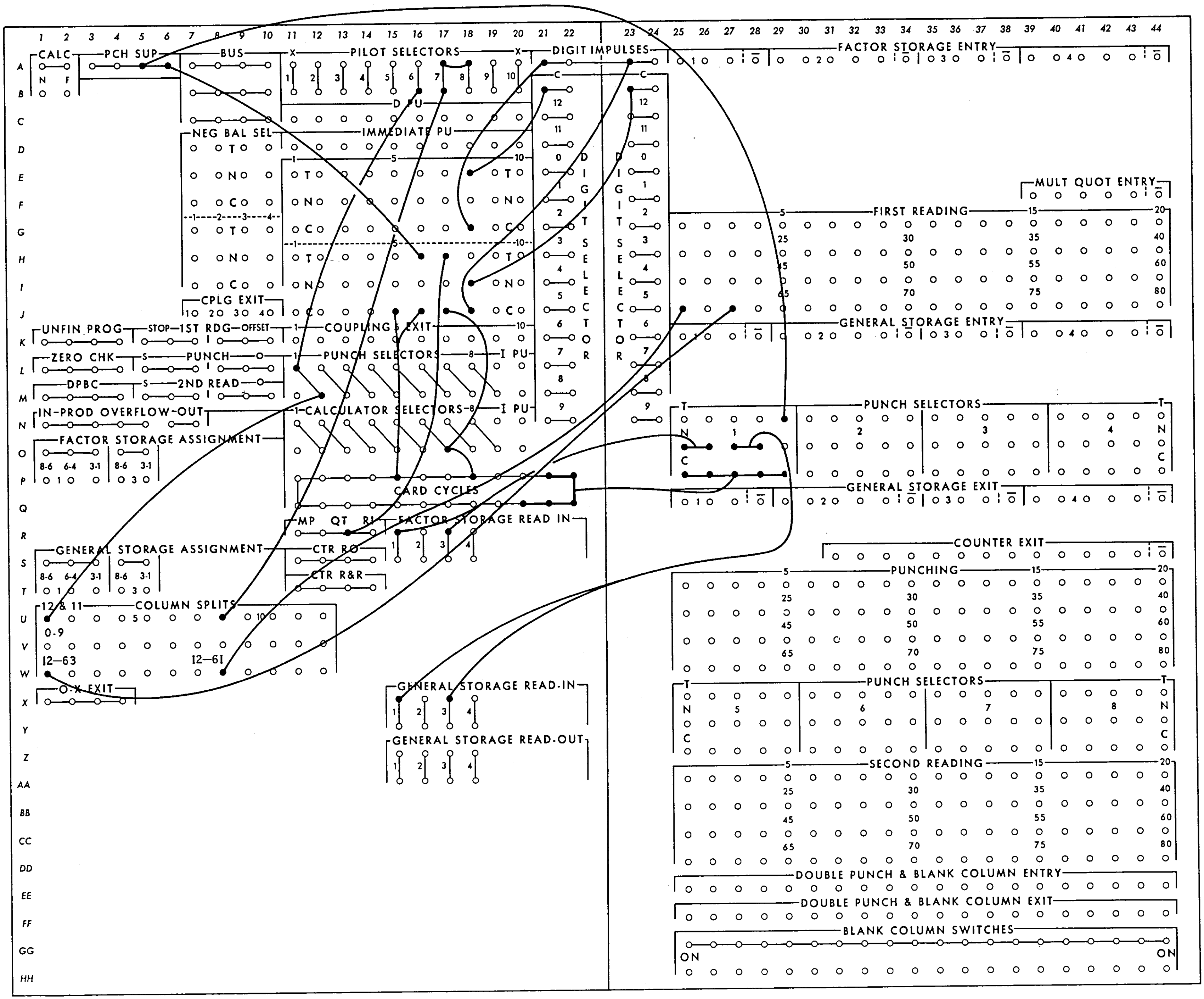
Other special wiring conditions:

Symbols: PS - Punch Selector; FS - Factor Storage Entry; GS - General Storage Entry
DS - Digit Selector; T - Transferred Side;
RI - Read in First Reading; N - Normal Side;
col. spl. - Column Split

Figure 5

INTERNATIONAL BUSINESS MACHINES CORPORATION

ELECTRONIC CALCULATING PUNCH, TYPE 604 CONTROL PANEL



Supplementary Wiring Required to Read Sums from Two Cards for Each Case

MACHINE TIME REQUIREMENTS

The time required for this method is composed of three basic components. First, a relatively constant amount of time will be required to wire and test boards; the second component is the time required for program advances between runs; and the third component is the time used by the machine to run through the required cards.

In our experience, two hours were required to wire and test the two sets of boards. In developing a rule of thumb for estimating calculating time, we shall define this time as the constant C.

The second component may be determined by noting that approximately 1 1/2 minutes are required for each program advance during the raw-score to standard-score operation, and approximately 2 1/2 minutes during the final computation. We shall define these constants as A and B, respectively.

The third component is determined by noting the machine capacity of 100 cards per minute.

From the preceding sections and above information we can write the following formula estimating the required machine time:

$$\text{Machine time} = n\left(\frac{2N+3}{100}\right) + An + r\left(\frac{4N+21}{100}\right) + Br + C, \qquad (6)$$

where A, B and C are defined above, N is the number of cases, n is the number of variables, and $r = n^2/16$, the number of machine runs.

For our example these values would be:

$$\begin{aligned} MT &= 40\left(\frac{103}{100}\right) + 60 + 100\left(\frac{221}{100}\right) + 250 + 120 \\ &= 41 + 60 + 221 + 250 + 120 \\ &= 692 \text{ minutes or 11 1/2 hours} \end{aligned}$$

SUMMARY

The present paper outlines a method of computing intercorrelation matrices on the 604 Electronic Calculating Punch by means of separate conversion of raw scores to standard scores.

Correlation coefficients are punched on individual result cards on which both row and column designations are also punched in the same operation. Correlations are checked by independent reverse calculation of each coefficient.

This method appears to be most economical where a large number of variables must be correlated for small size samples. Its major utility seems to be restricted for samples not exceeding 500 and from 20th to 80th order matrices.

REFERENCES

1. IBM Type 604 Electronic Calculating Punch Principles of Operation Manual, Form No. 22-5279.

2. K.W. Schaie and C.R. Strother, "Age changes in the Primary Mental Abilities in a Group of Superior Older People," American Psychologist, 10 (1955), 339 (Abstr.).

A METHOD OF CARD PROGRAMMING THE 604 FOR GENERAL ENGINEERING CALCULATIONS

R. Baldur

Dominion Engineering Works, Ltd.
Montreal, Canada

Introduction

The idea of using the IBM Type 604 Electronic Calculating Punch as a Card-Programmed Calculator is not new. However, the methods described previously (References 1 and 2) have only a limited application for engineering work, since they do not enable the evaluation of the frequently occurring trigonometric functions, and they have a limited storage capacity. The following factors influenced the design described in this article:

1. It was required to be able to evaluate the maximum number of the most common mathematical functions.
2. A reasonable storage capacity was necessary to enable easy manual collation and sorting of cards after each pass.
3. In order to limit the duplication of the program deck, a method of storage by gang-punching had to be avoided.
4. Double punching had to be reduced to a minimum to simplify the key-punching operation.

Owing to the limitations of the machine, some compromise had to be accepted. In particular, the design considered here is only capable of dealing with the four basic arithmetic operations, evaluating the square root, and natural and inverse circular functions. It is not possible to calculate the hyperbolic, exponential and logarithmic functions, but a table look-up operation is available, which can to some extent compensate for that shortcoming. Three storage locations are always available for retaining factors or intermediate results. But as many as five can be used if only addition, subtraction, multiplication and division is performed, and four when the square root is calculated. All punching is normally suppressed, except for the results, when four distinct numbers are available. Finally, the design of the cards is such that there is only one card which requires double punching in any table look-up operation.

In order to achieve all the above features it was necessary to sacrifice some of the accuracy of the calculations. In the worst case, only three digits are reliable, although in most cases four and five are correct. Consequently, the results are as good or better than obtained with a slide rule and, therefore, in the majority of cases, quite satisfactory for engineering work.

Card Layout and Programming Sheet

The programming sheet is a duplicate of the card layout except for the punching field (fig. 1). When planning for the electronic calculator, it is possible to use the programming sheet directly. When completed, it will also serve as a chart for the key-punch operator. The various fields in the card are allocated as follows:

Column 1 - 4	Card numbers indicating the sequence of a program deck.
Column 5 - 7	Special codes or any other indicative information which may be included with the results.
Column 8 - 11	Operating and transfer instructions which control the functions of the machine.
Column 12 - 18	Reading instructions for the basic storage locations.
Column 19 - 60	Fields of the seven five-digit storage locations, each of which includes a column to indicate a negative quantity.
Column 61 - 80	Punching fields for four five-digit results.

Storage Units

Factor storage and general storage locations are indicated by the letters F and G, respectively. In order to maintain uniformity, all storage units can accept only five digits and a sign. The basic storage units which are involved in all mathematical operations are factor storage 1-3, the multiplier-quotient unit, and the counter. For simplicity, the numbers contained in those locations will be denoted by F, M and C, respectively.

In order to read a number into electronic storage, it is necessary to have the number punched in the appropriate field together with an X-punch in the corresponding read column (12-18). If the number is negative, an X is punched in the column denoted by a minus sign. The column is left blank for positive numbers. This arrangement is used to avoid double punching.

The three general storage units are not involved in any calculations and can be used for storing factors and results. The contents of these locations are also available for punching.

Storage units F2 and F4 are used for some calculations. When not in use, however, they are available as temporary storage locations but their contents cannot be punched. In particular, during addition, subtraction, multiplication, and division both F2 and F4 can be used as intermediate storage locations. When square roots are evaluated, only F4 is available. Neither F2 nor F4 can be used for storage during the calculation of natural or inverse trigonometric functions, or during the table look-up operation.

Storage location F1-3 is the basic unit in which the numbers to be operated on are contained. When the operation involves two numbers, as in addition, subtraction, etc., the other factor must be located in M. In all operations except division, the number F, stored in F1-3, is retained throughout the calculation. At the end of any operation the five-digit result is available in M and C.

All storage positions retain their numbers until another number is read into them. The electronic counter, C, is the only exception; it is reset at the end of each card cycle.

Operating Instructions

Columns 8-11 control the operation of the machine. There are two main groups of instructions which can be given to the calculator. Columns 8 and 9 contain primarily the codes for arithmetic functions, whereas columns 10 and 11 indicate the transfer operations. In most cases the two functions are independent and can be performed simultaneously.

The details of the various codes are given below:

Arithmetic Functions

Addition. Code 1 in Column 8.

The operation performed is .1(F+M). Both factors are assumed to be of order .xxxxx and the results of order x.xxxx, correct to four decimal places.

Subtraction. Code 3 in column 8.

The operation performed is .1(F-M). Both factors are assumed to be of order .xxxxx, and the result of order x.xxxx, correct to four decimal places.

Multiplication. Code 2 in column 8.

The operation performed is F• M. Both factors are assumed to be of order .xxxxx, and the result of order .xxxxx, correct to five decimal places.

Division. Code 4 in column 8.

The operation performed is .1F/M. Both factors are assumed to be of order .xxxxx, and the result of order x.xxxx, correct to four decimal places. Should the result be greater than 9.9999, a division overflow will occur. This will cause the machine to stop and the zero check light to go on. The card which caused the division overflow will be at the top of the stacker.

Square Root. Code 7 in column 8.

The operation performed is $\sqrt{F}$. The number F and the result are both assumed to be of order x.xxxx. The maximum error can be ±.0001. If it is known that $0 \leqq F \leqq 1.0000$, then 10 cards are required in order to evaluate the square root to within the accuracy stated above. The iterative process is based on the formula

$$C_{n+1} = \frac{1}{2}(C_n + \frac{F}{C_n}),$$

where F is the number whose root is required and C_n is the nth approximation. In the above case the first approximation should be taken as 0.1001, which has to be read into F2 from the first card of the square root sequence. This should be followed by nine cards, each punched with a 7 in column 8. If the order of magnitude of F cannot be specified as above, the first approximation in F2 should be 1.0000 and followed by 13 iterations, making a total of 14 cards. At the end of the iteration the result is also available in F2, in addition to the usual locations M and C.

Cosine. Code 6 in column 8.

The operation performed is cos F. The angle F, in radians, is assumed to be positive, and of the order x.xxxx. The maximum value of F is π . The result is of the order x.xxxx, and at least correct to three decimal places, except for angles larger than about 150°. The iterative process is based on the formula

$$\cos F = 1 - \frac{F^2}{1.2} \{ 1 - \frac{F^2}{3.4}(1 - \frac{F^2}{5.6} \cdots)\}$$

Eight cards are required to obtain the accuracy given above. The numbers 01400 and 10000 should be read into M and F4, respectively, from the first card of the sequence. This should be followed by seven cards, each punched with a 6 in column 8. At the end of the iteration the result is also available in F4, in addition to the usual locations M and C. See Appendix 1 for calculation of other natural trigonometric functions and for calculations of cos F when F is negative.

Inverse Sine. Code 5 in column 8.

The operation performed is $1/F \sin^{-1} F$. The factor is assumed positive, of the order .xxxxx, and is limited by the conditions that $0 \leqq F \leqq 1/\sqrt{2}$. The results are of the order x.xxxx and the maximum error is .0001.

The iterative process is based on the formula

$$\frac{1}{F}\sin^{-1}F = 1 + \frac{1^2}{2.3}F^2\{1 + \frac{3^2}{4.5}F^2(1 + \frac{5^2}{6.7}F^2 \ldots)\}$$

Eight cards are required to obtain the accuracy given above. The numbers 01500, 00015, and 10000 should be read into M, F2, and F4, respectively, from the first card of the sequence. This should be followed by seven cards, each punched with a 5 in column 8. At the end of the iteration the result is also available in F4, in addition to the usual locations M and C. See Appendix 2 for calculation of other trigonometric functions and for calculation of $\sin^{-1}F/F$ when $1/\sqrt{2} \leqq F \leqq 1$ or $-1 \leqq F \leqq 0$.

Transfers

It is possible to transfer a factor from any storage unit into F1-3 before the calculations take place. Consequently, the transfer and the operation on that number can, in general, be performed during the same card cycle. The two exceptions are M and G4, which should not be used in calculations on the same card cycle as the transfer takes place. After each calculation, i.e., during the punching cycle, the results can either be punched or transferred from the counter into any one of the storage locations. Since at that time M and C contain the same number, the transfer between these two units is not required. It is also not possible to transfer from C to F1-3, but the transfer from M could be used instead. The numbers transferred from C are available on the following card cycle.

The transfer codes are given in detail below:

Transfer from F2 to F1-3. Code 4 in column 10.

Transfer from F4 to F1-3. Code 5 in column 10.

Transfer from G1-3 to F1-3. Code 1 in column 10.

Transfer from G2 to F1-3. Code 2 in column 10.

Transfer from G4 to F1-3. Code 3 in column 10.

The number transferred can be used for calculations on the same card cycle only if the shift code (code 9 in column 9) is not punched in the same card. If the shift code is present, the number in F1-3 will be shifted one position to the left before the arithmetic operation takes place. This feature may be of advantage in some cases.

Transfer from M to F1-3. Code 3 in column 10 together with code 0 in column 8. The remarks of the previous paragraph apply here as well. In addition, since this transfer requires punching in the operation column 8, double punching will be required in order to perform an arithmetic operation during the same card cycle.

Transfer from C to F2. Code 9 in column 11.

Transfer from C to F4. Code Y (12th position) in column 11.

Transfer from C to G1-3. Code 6 in column 11.

Transfer from C to G2. Code 7 in column 11.

Transfer from C to G4. Code 8 in column 11.

Transfer to F1-3 takes place at the beginning of the calculating time, while transfer from C occurs during punching time. Consequently, both codes can be used independently on the same card. If a code in column 11 is used, the transfer will take place when the card containing the code reaches the punching station.

Shift. Code 9 in column 9.

During addition, subtraction, and division the decimal point is shifted one place to the right. This is necessary, since the order of magnitude increases during those operations. When the numbers which are involved in the calculation are small, it may be desirable to maintain the decimal point at the same place. This can be done by shifting the result one position to the left. If a shift is used during multiplication, the order of magnitude is reduced by one.

The above can be summarized symbolically as follows:

Add and shift	.xxxxx + .xxxxx = .xxxxx
Subtract and shift	.xxxxx - .xxxxx = .xxxxx
Multiply and shift	.xxxxx × .xxxxx = .0xxxxx
Divide and shift	.xxxxx ÷ .xxxxx = .xxxxx

In the case of addition and subtraction the results are correct to five decimal places. Only five digits are available for the result of multiplication, and the first decimal place shown as zero does not appear in the result, which is correct to six decimal places. In the division, the last digit is not half adjusted, i.e., the maximum error is .00001. The result cannot be shifted during the evaluation of square roots, cosines, or inverse sines. The use of the shift code in that case will be ineffective. In some cases shift can be applied together with a transfer operation. This feature is normally used for transfer of numbers from M to F1-3, although the shift can also be made during transfer from G4 to F1-3. The above is equivalent to direct multiplication by ten.

Test for Sign of Arguments. Code Y (12th position) in column 8. Code 8 in column 9.

The calculation of cosine and inverse sine can only be made for positive arguments. It is therefore essential to be able to determine the sign of the argument before the evaluation of the function is started, and to change the number to its absolute value.

If code Y in column 8 is then used on the following card cycle, the factor contained in M will be changed to its absolute value. Moreover, if the original value of M was negative, one of the pilot selectors will be picked up and remain transferred until a special code is used to bring it back to normal. The latter is code 8 in column 9, which can be used on any subsequent card. This code also enables the subtraction F-M to be performed on the following card cycle if the original argument in M was negative; otherwise no operation takes place.

To illustrate the above, consider the calculation of $\cos^{-1}A$. As a preliminary step the argument A is transferred to M and tested for its sign by means of code Y in column 8. On the next card cycle the value $|A|$ will be given in M. A series of cards will now follow in order to perform the calculation which is only possible for positive arguments (see Appendix 2). At the end of this process the result, say X, is again inserted in M, and the value π is read into F1-3. The code 8 in column 9 is then given. If originally A was negative, subtraction will be performed on the next card cycle, giving $(\pi - X)$ as a result; if A was positive, no operation will take place and. X will remain as the result. When code 8 in column 9 is used, it is assumed that both numbers M and F are of order .xxxxx and the result is also of the same order, correct to five decimal places.

The test for negative arguments can also be made before the extraction of a square root. In this case a test division could be programmed in such a way as to cause overflow if the factor was negative. As a result the machine will stop, indicating that the square root operation was invalid.

The only limitation for the above test is that no result may be punched on any intermediate card until the particular calculation is completed. In other words, there should be no card containing code X in column 11 (punching code) between the card having code Y in column 8 and the card with code 8 in column 9. It is unlikely that such a case would be required in practice.

Punching. Code X in column 11.

All punching is done on blank cards. Consequently, the code which actuates the operation should be given on the preceding card. When code X in column 11 is given, the contents of G1-3, G2, G4, and C are punched into their respective fields on the following card.

Sometimes it may be necessary to include identifying codes together with the punched results. This may be done by gang punching into the special codes from the preceding or succeeding card, depending whether the gang punching is wired from second or first reading. If several results are interspersed in the same deck, it may be convenient to gang punch the card number of one of the adjacent cards. During the punching operation factors cannot be transferred from C into any of the storage locations.

Table Look-Up. Code X and 3 in column 8 together with codes 8 and 9 in column 9.
Code X in column 8 together with code 8 in column 9.

The table look-up has two distinct types of cards. The first card of each table prepares the machine for the table look-up operation. This card has codes X and 3 in column 8 together with codes 8 and 9 in column 9. The remaining cards contain the actual table and have code X in column 8 together with code 8 in column 9. Before the start of the operation the searching argument is placed in F1-3. The layout of the table cards is as follows: the arguments are in field M, the function in field F2, and the first order differences in field F4. During the operation, the argument in M is subtracted from the searching argument in F1-3 and the fields F2 and F4 are read in. If the difference between the arguments is negative, the operation is repeated for the following card and the process goes on until a zero or positive difference is obtained. Subsequently, the remaining cards of the table pass through without being read and subtraction is suppressed. Consequently, at the end of the table look-up the searching argument is in F1-3; storage unit M contains the difference between the searching argument and the first argument of the table which is smaller or equal to it; F2 and F4 contain, respectively, the function and first order difference for the latter argument.

The table must always be arranged in a descending order of arguments. No reading instructions (columns 12-18) should be included in the table cards. In order to prepare the machine for the table look-up, it is necessary to perform a subtraction which gives a negative result. This is obtained by reading the number 99999 into M. Consequently, the first table card should consist of the appropriate codes (X-3 and 8-9)99999 in field M, reading instructions for M (column 12), and possibly the searching arguments and reading instructions in field F1-3. The subtraction during the table look-up is of the type F-M or .xxxxx - .xxxxx = .xxxxx, correct to five decimal places. The table look-up is possible for positive arguments only.

Wiring Diagrams

The wiring diagrams are shown in various sections for ease of reading. Figures 2, 3, and 4 show the wiring of different parts of the punch panel. In a few cases the same split-wire is given on more than one diagram. The connections for the calculator selectors on the calculate control panel are given in symbolic form in figure 5. The meaning of the symbols is as follows:

㉓ A number enclosed by a circle - suppression hub of a particular program step.

Ⓒ	A letter enclosed by a circle - suppression hubs of a suppression group.
$\boxed{37}$	A number enclosed by a rectangle - one program outlet of a particular program step.
S	Suppress without balance test.
SUP +	Suppress on plus balance.
SUP -	Suppress on minus balance.
BTSS	Balance test for step suppression.
BTSPU	Balance test for selector pickup.
RI, RO	followed by the symbol of a storage unit denotes read into, and read out of the indicated storage unit.
RI +	Read into counter +
RI -	Read into counter -
M +	Multiply +
M -	Multiply -
DIV	Divide
Z.CH.	Zero check
1/2	Half adjust
RUI, RUO	followed by a number denotes read unit into, and read unit out of the position indicated.
ET	followed by a number denotes number obtained from emitter control.
RO	Read out of counter
R&R	Read out and reset counter

The individual program steps which are a guide to the wiring of the calculate panel are given in tabular form in figures 6(a) and 6(b). Since depending on the operation codes, etc., different operations may be performed on the same program step, all the possible combinations are included in the table. Each line indicates the codes corresponding to the particular operation steps and the calculator selectors which are transferred at that time. The program steps are combined into a number of suppression groups, which are referred to in figure 4. Suppression group B is different from all others and is indicated by *. The three program steps included in that group are always operative, except when calculator selector 3 is transferred, i.e., either when code 3 or code 5 are used in column 8.

The wiring of the punch panel presented considerable difficulties, and it was not possible to avoid all the back circuits owing to a shortage of selectors. Consequently, three small rectifiers had to be added and these are indicated by the standard symbol. Two types of rectifiers were tried and both gave satisfactory results: the selenium type, consisting of three plates, each one inch square; and germanium diodes, only about 3/8″long and 3/16″ in diameter.

Another effect of the shortage of pilot selectors is some unusual features used in the wiring. In particular, both double punch detection and product overflow are used as delayed pickups.

The programming is such that there is only just enough time to perform all the calculations during one card cycle and the unfinished program is wired to stop the machine as a safety precaution.

Appendix 1

The calculator is capable only of evaluating cosines of positive angles. If other trigonometric functions are required, they can be obtained from the following formulae:

$$\sin X = \sqrt{1-\cos^2 X}$$
$$\tan X = \frac{\sin X}{\cos X}$$

In order to ensure that the iteration is performed for positive arguments only, the absolute value of the argument should be obtained as a first step. The required function is then calculated using the above formulae. At the end of the calculation, one of the following subtractions is performed if the factors were initially negative:

$$0 - \sin X$$
$$0 - \tan X$$
$$2 \cos X - \cos X$$

Appendix 2

A large number of iterations is required to obtain $1/X \sin^{-1} X$ accurately for X close to unity. In order to reduce the calculating time, the value of X has been limited to $1/\sqrt{2}$, corresponding to an angle of $\pi/4$ or 45°.

Let Y be any positive argument such that

$$0 \leqq Y \leqq 1$$

and let $\sin^{-1} Y = A = \frac{\pi}{4} + B$

If B is now considered as the unknown quantity it follows that

$$-\frac{\pi}{4} \leqq B \leqq \frac{\pi}{4}$$

Let $B = \sin^{-1} X$

then $-\frac{1}{\sqrt{2}} \leqq X \leqq \frac{1}{\sqrt{2}}$

The relationship between X and Y is as follows for the different inverse trigonometric functions:

To obtain	$A = \sin^{-1} Y$	for $0 \leqq Y \leqq 1$
find	$X = \frac{1}{\sqrt{2}} \left(Y - \sqrt{1-Y^2}\right)$	
giving	$B = \sin^{-1} X$	
and	$A = \frac{\pi}{4} + B$	
To obtain	$A = \cos^{-1} Y$	for $0 \leqq Y \leqq 1$
find	$X = \frac{1}{\sqrt{2}} \left(Y - \sqrt{1-Y^2}\right)$	
giving	$B = \sin^{-1} X$	
and	$A = \frac{\pi}{4} - B$	
To obtain	$A = \tan^{-1} Y$	for $0 \leqq Y \leqq 9.8$
find	$X = \frac{1}{\sqrt{2}} \frac{X-1}{\sqrt{X^2+1}}$	
giving	$B = \sin^{-1} X$	
and	$A = \frac{\pi}{4} + B$	

Due to the limited calculator capacity, the maximum value of $\tan^{-1} Y$ has to be limited to just over 84°.

To ensure that positive arguments only are used, the procedure is the same as that given in Appendix 1, except that one of the following subtractions is performed at the end:

$$0 - \sin^{-1} Y$$

$$\pi - \cos^{-1} Y$$

$$\pi - \tan^{-1} Y$$

The above subtractions can be modified if different quadrants are required.

Appendix 3

For most calculations it is arbitrarily assumed that the numbers are less than unity and the various powers of 10 are obtained by shifts to the left or right. In the case of some operations, however, it is important to locate the decimal point at the place determined by the programming. It is therefore suggested that the following scheme be used throughout.

.xxxxx + .xxxxx = x.xxxx	without shift
= .xxxxx	with shift
.xxxxx - .xxxxx = x.xxxx	without shift

	= .xxxxx	with shift
.xxxxx X .xxxxx	= .xxxxx	without shift
	= .0xxxxx	with shift
.xxxxx ÷ .xxxxx	= x.xxxx	without shift
	= .xxxxx	with shift
$\sqrt{x.xxxx}$	= x.xxxx	
cos (x.xxxx)	= x.xxxx	
$\frac{1}{(.xxxxx)} \sin^{-1}(.xxxxx)$	= x.xxxx	

References

1. Nims, P. T. "The IBM Type 604 Electronic Calculating Punch as a Miniature Card-Programmed Electronic Calculator," Proceedings, Computation Seminar, August, 1951 (IBM), pp. 37-47.

2. Wall, D. D. "A New Method for Card Programming the 604" (IBM), May, 1952.

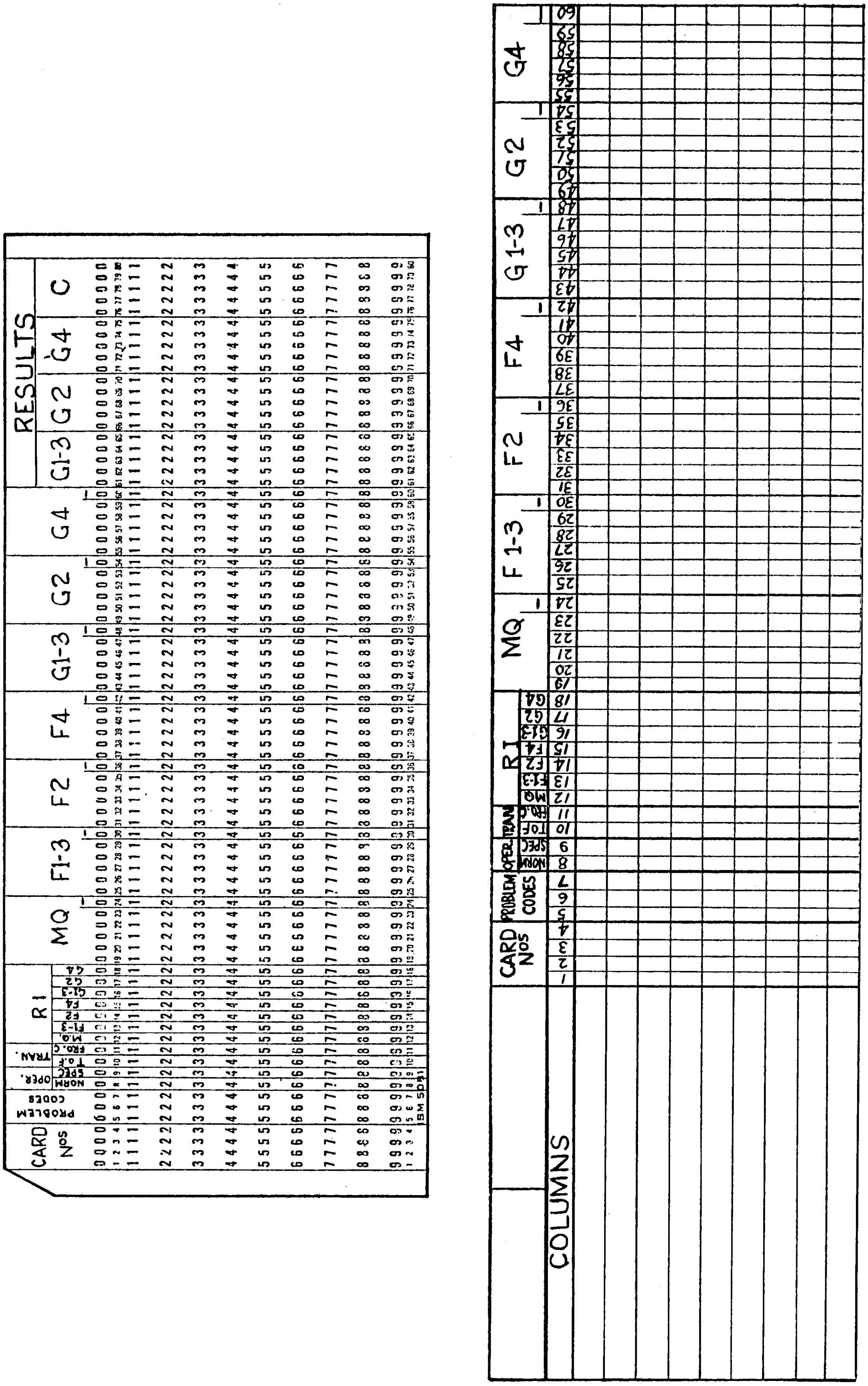

CARD Nos
PROBLEM CODES
OPER. NORM SPEC
TRAN. T.o.F FR0.C
R1 M.Q. F1-3 F2 F4 G1-3 G2 G4
MQ
F1-3
F2
F4
G1-3
G2
G4
RESULTS G1-3 G2 G4 C
IBM 5081
COLUMNS
CARD Nos 1 2 3 4
PROBLEM CODES 5 6 7
OPER. NORM 8 SPEC 9
TRAN. T.o.F 10 FR0.C 11
R1 MQ 12 F1-3 13 F2 14 F4 15 G1-3 16 G2 17 G4 18
MQ 19 20 21 22 23 24
F 1-3 25 26 27 28 29 30
F2 31 32 33 34 35 36
F4 37 38 39 40 41 42
G 1-3 43 44 45 46 47 48
G2 49 50 51 52 53 54
G4 55 56 57 58 59 60

Figure 1

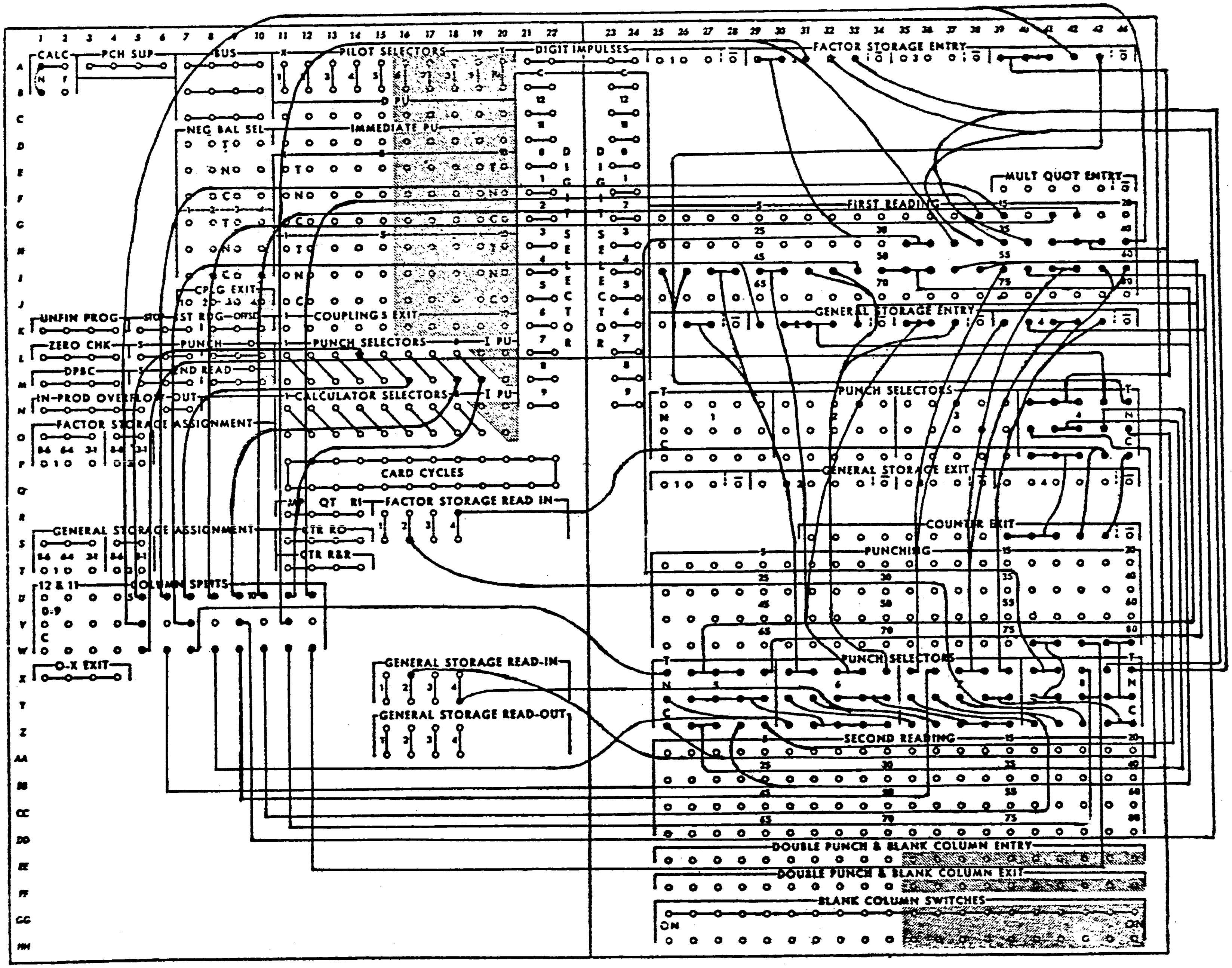

CALC
PCH SUP
BUS
PILOT SELECTORS
DIGIT IMPULSES
FACTOR STORAGE ENTRY
NEG BAL SEL
IMMEDIATE PU
MULT QUOT ENTRY
FIRST READING
DIGIT SELECTOR
CPLG EXIT
UNFIN PROG
ST PROG
COUPLINGS EXIT
GENERAL STORAGE ENTRY
ZERO CHK
PUNCH
PUNCH SELECTORS
DPBC
END READ
IN-PROD OVERFLOW-OUT
CALCULATOR SELECTORS
FACTOR STORAGE ASSIGNMENT
CARD CYCLES
GENERAL STORAGE EXIT
QT
RI
FACTOR STORAGE READ IN
GENERAL STORAGE ASSIGNMENT
CTR RO
CTR R&R
COUNTER EXIT
PUNCHING
12 & 11
0-9
COLUMN SPLITS
O-X EXIT
GENERAL STORAGE READ-IN
GENERAL STORAGE READ-OUT
PUNCH SELECTORS
SECOND READING
DOUBLE PUNCH & BLANK COLUMN ENTRY
DOUBLE PUNCH & BLANK COLUMN EXIT
BLANK COLUMN SWITCHES
ON

Figure 2

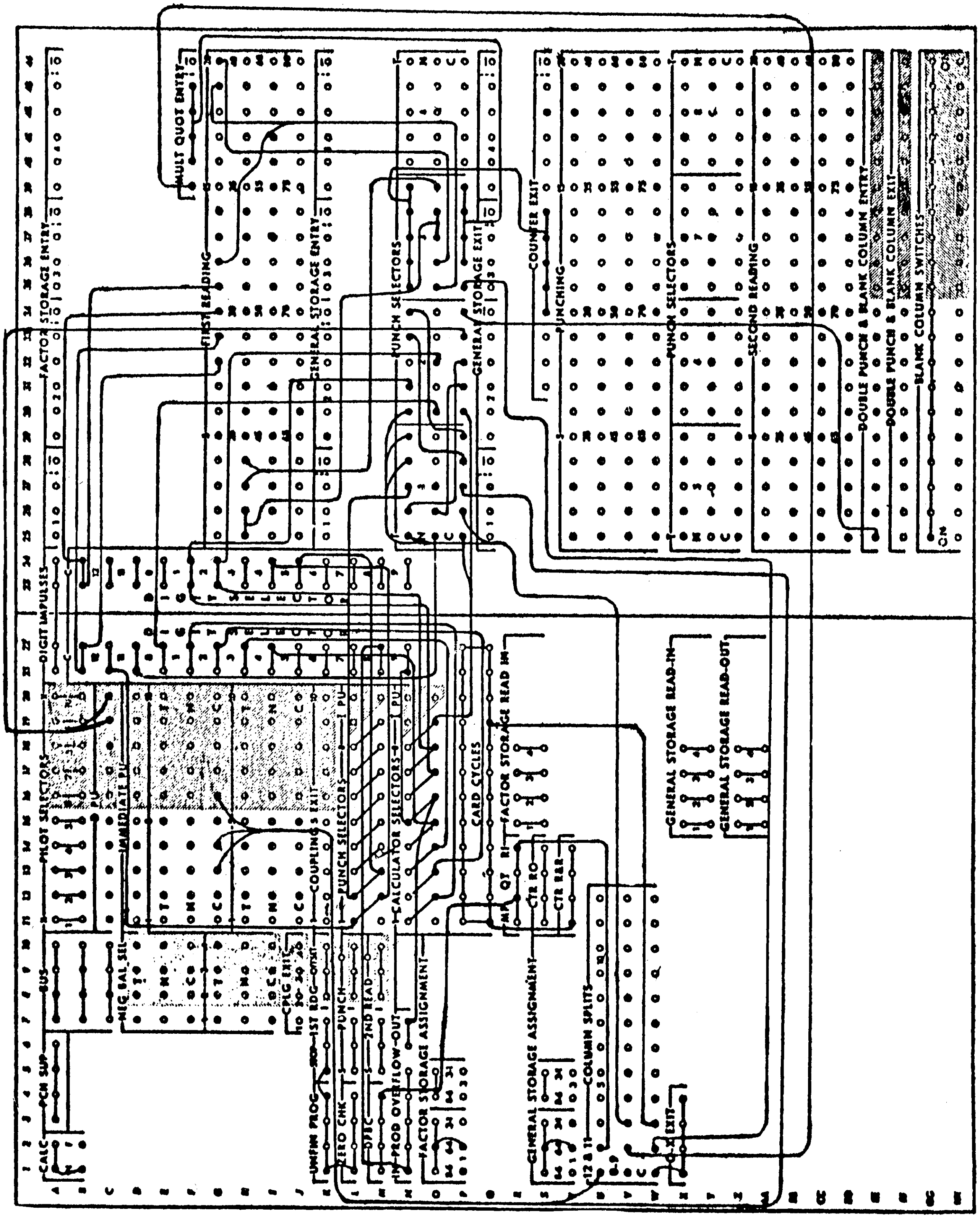

CALC
PCH SUP
BUS
NEG BAL SEL
PILOT SELECTORS
IMMEDIATE PU
DIGIT IMPULSES
CPLG EXIT
COUPLINGS EXIT
PUNCH SELECTORS
CALCULATOR SELECTORS
CARD CYCLES
FACTOR STORAGE READ IN
GENERAL STORAGE READ-IN
GENERAL STORAGE READ-OUT
UNFIN PROG
ZERO CHK
1ST RDG
PUNCH
2ND READ
DPBC
PROD OVERFLOW-OUT
FACTOR STORAGE ASSIGNMENT
GENERAL STORAGE ASSIGNMENT
COLUMN SPLITS
EXIT
MP
QT
RI
CTR RO
CTR RRR
MULT QUOT ENTRY
FACTOR STORAGE ENTRY
FIRST READING
GENERAL STORAGE ENTRY
PUNCH SELECTORS
GENERAL STORAGE EXIT
COUNTER EXIT
PUNCHING
PUNCH SELECTORS
SECOND READING
DOUBLE PUNCH & BLANK COLUMN ENTRY
DOUBLE PUNCH & BLANK COLUMN EXIT
BLANK COLUMN SWITCHES

Figure 3

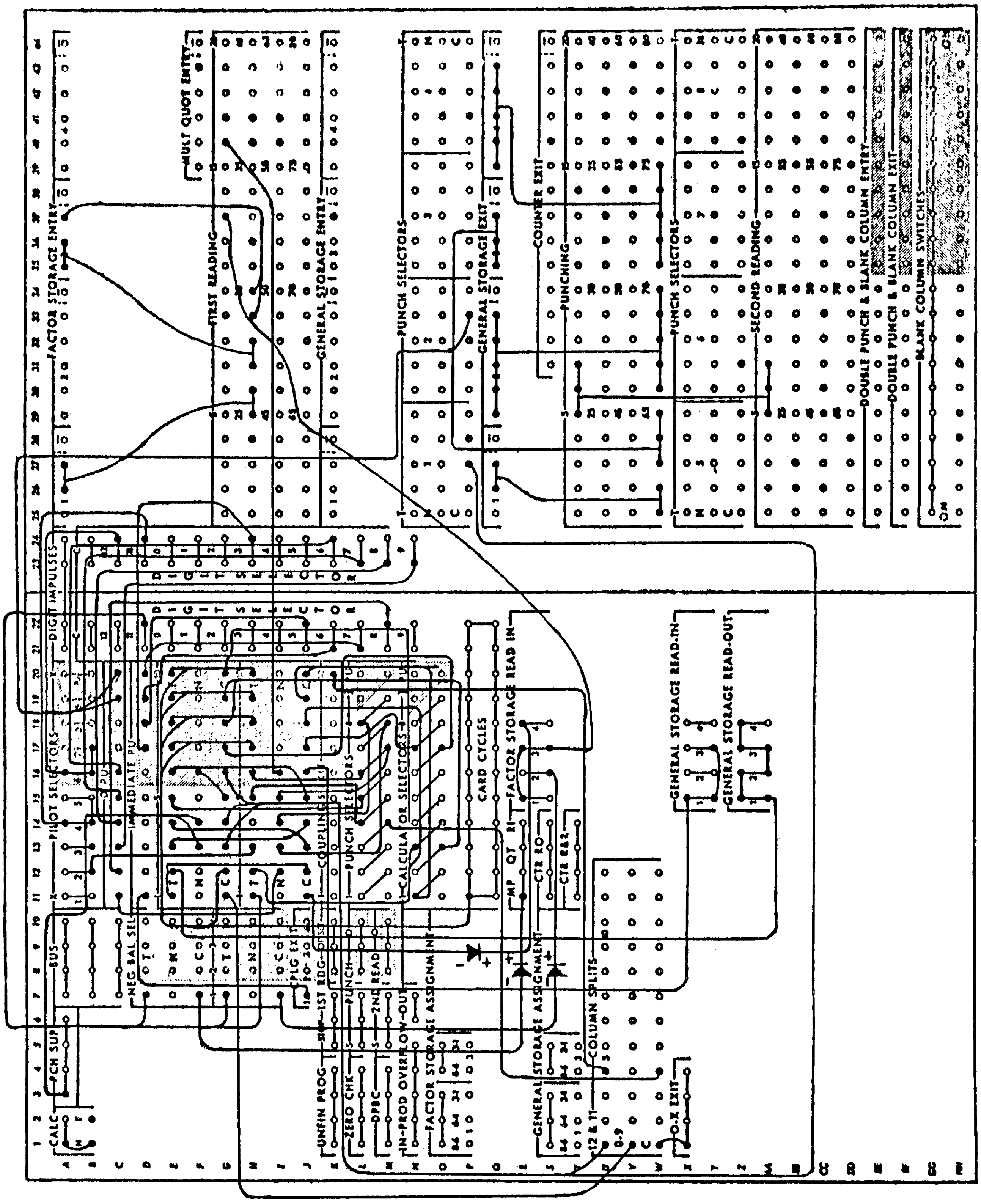
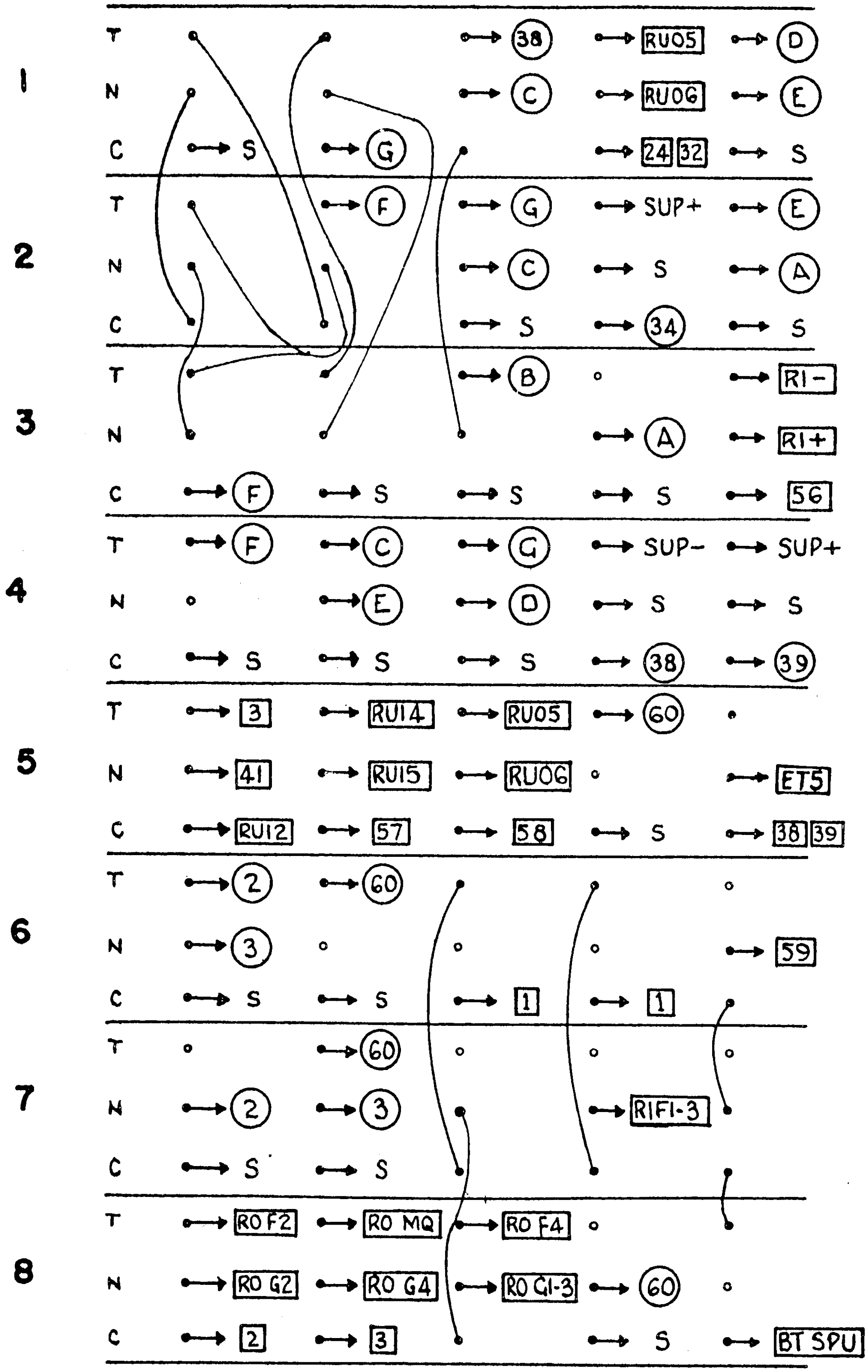

CALC
PCH SUP
BUS
NEG BAL SEL
PILOT SELECTORS
IMMEDIATE PU
DPU
DIGIT IMPULSES
DIGIT SELECTOR
FACTOR STORAGE ENTRY
FIRST READING
MULT QUOT ENTRY
GENERAL STORAGE ENTRY
PUNCH SELECTORS
GENERAL STORAGE EXIT
COUNTER EXIT
PUNCHING
SECOND READING
DOUBLE PUNCH & BLANK COLUMN ENTRY
DOUBLE PUNCH & BLANK COLUMN EXIT
BLANK COLUMN SWITCHES
COUPLINGS
PUNCH SELECTORS
CALCULATOR SELECTORS
CARD CYCLES
FACTOR STORAGE READ IN
MP
QT
RI
CTR RO
CTR RRR
GENERAL STORAGE READ-IN
GENERAL STORAGE READ-OUT
SPLG EXIT
1ST RDG
PUNCH
2ND READ
UNFIN PROG
ZERO CHK
DPBC
IN-PROD OVERFLOW-OUT
FACTOR STORAGE ASSIGNMENT
GENERAL STORAGE ASSIGNMENT
COLUMN SPLITS
12 & 11
0-9
0-X EXIT

Figure 4



CALCULATE SELECTORS

Figure 5

PROC. NO.	OPERATION			SUP. ON	SUP. GR	CALCULATOR SELECTORS								OPERATION CODES IN COLUMNS		
						1	2	3	4	5	6	7	8	8	9	10
1			BT SS													
	RI G1-3	RI F1-3	BT SS								T					1
	RO F4	RI F1-3	BT SS								T		T			5
2	RO G2	RI F1-3										T				2
	RO F2	RI F1-3										T	T			4
3	RO G4	RI F1-3									T	T				3
	RO G4	RI F1-3	RUI 2							T	T	T			9	3
	RO MQ	RI F1-3									T	T	T	0		3
	RO MQ	RI F1-3	RUI 2							T	T	T	T	0	9	3
4	RO F2	M+			A		T	T						5		
5	R&R	RI F2			A		T	T						5		
6	RO F2	RI +	RUI 6		A		T	T						5		
7	RO MQ	RI +			C	T	T							6		
							T	T						5		
8	ET 1	RI +	RUI 3		C	T	T							6		
							T	T						5		
9	RO	RI F2			C	T	T							6		
							T	T						5		
10	ET 1	RI+	RUI 3		C	T	T							6		
							T	T						5		
11	RO	RI MQ	RUO 3		C	T	T							6		
							T	T						5		
12	RO MQ	RI −	RUI 3		C	T	T							6		
							T	T						5		
13	RO F2	M +			C	T	T							6		
							T	T						5		
14	RO	RI F2			C	T	T							6		
							T	T						5		
15	RO F2	RI −			C	T	T							6		
							T	T						5		
16	RO MQ	RI +			C	T	T							6		
							T	T						5		
17	RO F1-3	RI +	RUI 3		B			*						*		
18	RO F2	DIV			C	T	T							6		
							T	T						5		
19	RO	RI F2	RUO 3		C	T	T							6		
							T	T						5		
20	RO F2	RI −	RUI 3		C	T	T							6		
							T	T						5		
21	ET 4	RI −	BT SS		C	T	T							6		
							T	T						5		
22	R&R	RI F2			C	T	T							6		
							T	T						5		
23	RO F1-3	M +			C	T	T							6		
							T	T						5		
24	R&R	RI MQ	RUD 5		C	T	T							6		
			RUO 6		C		T	T						5		
25	RO F4	M −			B			*						*		
26	RO F1-3	M +			A		T	T						5		
27	R&R	RI MQ	RUO 6		A		T	T						5		
28	RO F4	M+			A		T	T						5		
29	ET 1	RI F4	RUI 5		C	T	T							6		
							T	T						5		
30	RO F4	RI +	RUI 6		A		T	T						5		
31	RO F4	RI +	RUI 5		B			*						*		

Figure 6a

PROG. NO	OPERATION			SUP. ON.	SUP. GR.	CALCULATOR SELECTORS.								OPERATION CODES IN COLUMNS		
						1	2	3	4	5	6	7	8	8	9	10
32	R&R	RI F4	RUO 5		C	T	T							6		
			RUO 6				T	T						5		
33	RO F2	RI MQ	RUI 3		C	T	T							6		
							T	T						5		
34	RO F4	RI MQ		+		T	T							6		
							T	T						5		
35	R&R															
36	RO F1-3	M+	BT SS		D				T					4		
37	R&R				D				T					4		
38	M+	ET 5		−					T					4		
	M+			−					T	T				4	9	
38	M−	ET 5		+					T					4		
	M−			+					T	T				4	9	
40	RO F1-3	RI +	RU 16		D				T					4		
41	RO MQ	RI F1-3	RU 12		D				T					4		
									T	T				4	9	
42	RO F1-3	DIV			D				T					4		
43	R&R	RI F1-3	RUO 6		D				T					4		
44	RO F1-3	RI +			D				T					4		
45	Z.CH.	RUI 2			D				T					4		
46	RO F1-3	RI +	RUI 5		E	T			T					7		
47	RO F2	DIV			E	T			T					7		
48	R&R				E	T			T					7		
49	RO F2	RI +			E	T			T					7		
50	RO MQ	RI +			E	T			T					7		
51	ET 2	DIV			E	T			T					7		
52	RO MQ	RI F2			E	T			T					7		
53	RO F1-3	M+			F	T								1		
							T							2		
								T						3		
54	R&R				G	T								1		
								T						3		
55	RO F1-3	RI +	RU 15		G	T								1		
								T						3		
56	RO MQ	RUI 5	RI +		G	T								1		
			RI −					T						3		
57	$\frac{1}{2}$	RU15			F	T								1		
							T							2		
								T						3		
	$\frac{1}{2}$	RU 14				T				T				1	9	
							T			T				2	9	
								T		T				3	9	
58	R&R	RI MQ	RUO 6		F	T								1		
							T							2		
								T						3		
	R&R	RI MQ	RUO 5			T				T				1	9	
							T			T				2	9	
								T		T				3	9	
59	RO MQ	RI +														
			BT SPU										T	Y		
			BT SPU					T		T			T	X/3	8/9	
			BT SPU					T		T			T	X	8	
			BT SPU							T			T	X	8	
60	RO MQ	RI +	RUI 6										T	Y		

Figure 6b

NUMERICAL INTEGRATION OF HYPERBOLIC PARTIAL DIFFERENTIAL EQUATIONS WITH CONSTANT COEFFICIENTS AND WITH CHARACTERISTIC INITIAL CONDITIONS ON THE 604 ELECTRONIC CALCULATING PUNCH

N. Arne Lindberger

International Business Machines Svenska AB
Stockholm

This paper is concerned with the numerical solution by means of punch cards of the equation

$$\frac{\partial^2 \phi}{\partial \xi \, \partial \eta} + a \frac{\partial \phi}{\partial \xi} + b \frac{\partial \phi}{\partial \eta} + c\phi = 0 ,$$

where a, b, and c are constants. The initial values of ϕ are given along two orthogonal characteristics.

In section 1.1, by transformations in the dependent and the independent variables, the equation is reduced to the form

$$\frac{\partial^2 z}{\partial x \, \partial y} = z$$

In section 2.1 by approximation to a difference equation, a step-by-step procedure is derived, whose error and stability properties are analyzed in sections 2.2 and 3.1. On the basis of these results, section 4.1 describes a modification of the step-ahead process which has been set up for automatic integration on the 604 for the particular starting conditions $z(x,0) = 0$; $z(0,y) = e^y-1$. The corresponding values of ϕ have also been calculated for the case of a=b=1, c=0.

The machine set-up is described in sections 6 and 7.

In sections 5.1 and 5.2 the general solution of the problem is given as two different series expansions which have been used for an accuracy check on the results of the 604 runs.

1.1 By means of a linear transformation in the independent variables, any linear homogenous hyperbolic differential equation with constant coefficients can be reduced to the canonical form[1]

$$\frac{\partial^2 \phi}{\partial \xi \, \partial \eta} + a \frac{\partial \phi}{\partial \xi} + b \frac{\partial \phi}{\partial \eta} + c\phi = 0, \qquad (1.11)$$

where a, b, and c are constants.

The boundary conditions are assumed to be given along two characteristic curves, belonging to different sets; that is, ξ and η equal constants.

The partial derivatives of the first order can be removed from equation (1.11) by the transformation[2]

$$\phi(\xi,\eta) = z(\xi,\eta)\cdot e^{-b\xi - a\eta} \tag{1.12}$$

Equation (1.11) then reduces to

$$\frac{\partial^2 z}{\partial\xi\,\partial\eta} = (ab-c)z \tag{1.13}$$

The trivial case of c=ab will be disregarded.

At last $$x = \frac{ab-c}{K}\cdot\xi \;;\quad y = K\eta\,, \tag{1.14}$$

where K is an arbitrary constant, will transform (1.13) into

$$\frac{\partial^2 z}{\partial x\,\partial y} = z\,, \tag{1.15}$$

which is the form that is going to be treated numerically.

1.2 Formal integration of (1.15) over the domain indicated in Figure 1 yields[3]

$$z_M = z_P + z_Q - z_0 + \int_0^x\int_0^y z(\alpha,\beta)\,d\alpha\,d\beta\,, \tag{1.21}$$

where α, β are integration variables.

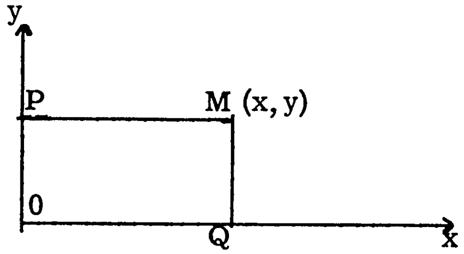


Figure 1

2.1 For an approximation to finite variables the first quadrant of the xy-plane is covered with a net of equal rectangles by the straight lines

$$x = n\,\Delta x,\quad n = 0, 1, 2, 3, \ldots$$
$$y = m\Delta y,\quad m = 0, 1, 2, 3, \ldots$$

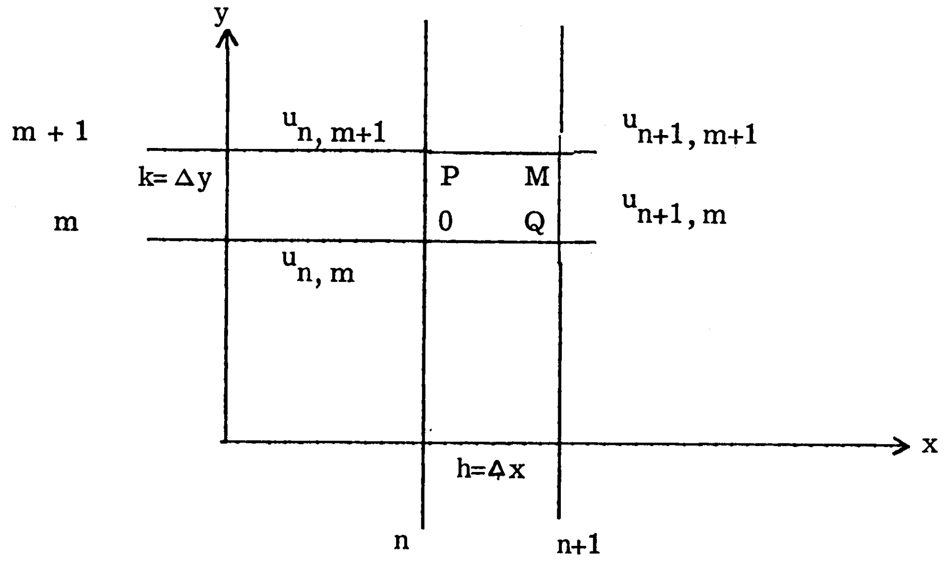


Figure 2

The equation (1.21) is now applied on a function $u_{n,m}$ defined at the nodes of the mesh and intended to be an approximation to z(x, y). The integral in (1.21) is approximated by

$$\frac{hk}{4} \left(u_{n+1,m+1} + u_{n+1,m} + u_{n,m+1} + u_{n,m} \right) , \qquad (2.11)$$

where $\quad h = \Delta x, \quad k = \Delta\, y, \quad u_{n,m} \simeq z(x, y)$,

i.e., the integrand is treated as a constant which is equal to the mean of the nodal values.

Insertion into equation (1.21) now yields

$$u_{n+1,m+1} = (u_{n+1,m} + u_{n,m+1}) \, (1+ \delta) - u_{n,m} \qquad (2.12)$$

$$\delta = \frac{hk/2}{1-hk/4} \qquad (2.13)$$

Incidentally, this formula is not only a first approximation: it also represents the limes of a Picard's iteration process^4 (see section 5.1) on equation (1.21) with the integral approximation (2.11).

Equation (2.12) is the equivalent of (1.15) in the finite difference domain. Of course, other approximations could have been obtained by other means of approximating the integral. Before applying equation (2.12) to numerical integration, its error properties will be investigated from various aspects.

2.2 The function $u_{n,m}$ defined by (2.12) and (2.13) is supposed to be an approximation of the function z(x, y) defined by (1.15). When inserting z into (2.12), it will be necessary to add an error quantity t(x+h, y+k) to the left-hand side

$$z_{n+1,m+1} + t_{n+1,m+1} = (z_{n+1,m} + z_{n,m+1}) (1+\delta) - z_{n,m} \qquad (2.21)$$

By using the Taylor's series for

$$z_{n+1,m} \equiv z(x+h,\ y) \text{ and } z_{n,m+1} \equiv z(x,y+k)$$

and expanding δ of (2.13) in a geometrical series, the right-hand side of (2.21) will be

$$\left(1 + \frac{hk}{2} + \frac{h^2\,k^2}{8} + \ldots\right) \left(z + h\frac{\partial z}{\partial x} + \frac{h^2}{\underline{\lfloor 2}}\ \frac{\partial^2 z}{\partial x^2} + \right.$$

$$+ \frac{h^3}{\underline{\lfloor 3}}\ \frac{\partial^3 z}{\partial x^3} + \frac{h^4}{\underline{\lfloor 4}}\ \frac{\partial^4 z}{\partial x^4} + \ldots .\ + z + k\frac{\partial z}{\partial y} + \frac{k^2}{\underline{\lfloor 2}}\ \frac{\partial^2 z}{\partial y^2} +$$

$$\left. + \frac{k^3}{\underline{\lfloor 3}}\ \frac{\partial^3 z}{\partial y^3} + \frac{k^4}{\underline{\lfloor 4}}\ \frac{\partial^4 z}{\partial y^4} + \ldots\right)_{n,m} - z_{n,m} \qquad (2.22)$$

When multiplying with hk/2, according to its definition in equation (1.15), z shall be regarded as $\partial^2 z/\partial x\partial y$. The same is valid for the partial derivatives of z. When multiplying with $h^2k^2/8$, z is set equal to $\partial^4 z/\partial^2 x\ \partial^2 y$.

With these rules, the expansion of (2.22) gives

$$z_{n,m} + h\ \frac{\partial z_{n,m}}{\partial x} + k\ \frac{\partial z_{n,m}}{\partial y} + \frac{1}{\underline{\lfloor 2}}\ \{ h^2\ \frac{\partial^2 z_{n,m}}{\partial x^2} + 2\,hk\ \frac{\partial^2 z_{n,m}}{\partial x\,\partial y} +$$

$$+ k^2\ \frac{\partial^2 z_{n,m}}{\partial y^2}\ \} + \frac{1}{\underline{\lfloor 3}}\ \{ h^3\ \frac{\partial^3 z_{n,m}}{\partial x^3} + 3\,h^2k\ \frac{\partial^3 z_{n,m}}{\partial x^2\,\partial y} +$$

$$+ 3\ hk^2\ \frac{\partial^3 z_{n,m}}{\partial x\partial y^2} + k^3\ \frac{\partial^3 z_{n,m}}{\partial y^3}\ \} + \frac{1}{\underline{\lfloor 4}}\ \{\ h^4\ \frac{\partial^4 z_{n,m}}{\partial x^4} +$$

$$+ 4\ h^3k\ \frac{\partial^4 z_{n,m}}{\partial x^3\,\partial y} + 6\ h^2k^2\ \frac{\partial^4 z_{n,m}}{\partial x^2\,\partial y^2} + 4\ hk^3\ \frac{\partial^4 z_{n,m}}{\partial x\,\partial y^3} + k^4\ \frac{\partial^4 z_{n,m}}{\partial y^4}\ \} +$$

$$+ \frac{hk}{12}\ (h^2\ \frac{\partial^4 z_{n,m}}{\partial x^3\,\partial y} + k^2\ \frac{\partial^4 z_{n,m}}{\partial x\,\partial y^3}) \qquad (2.23)$$

This is nothing but the Taylor series expansion of the left-hand side $z_{n+1,\ m+1} = z(x+\Delta x,\ y+\Delta y)$ plus an error term of the fourth order. Thus it has been shown that when using the recurrence formula (2.12), (2.13) for computing z, defined by (1.15), in each step of the process a truncation error

$$t\,(x,y) = \frac{\Delta x\ \Delta y}{12}\ \{\ (\Delta x)^2\ \frac{\partial^2 z}{\partial x^2} + (\Delta y)^2\ \frac{\partial^2 z}{\partial y^2}\ \} \qquad (2.24)$$

is committed.

3.1 After establishing the degree of approximation to the partial differential equation when using the difference equation (2.12), its stability properties will be investigated. Using a method attributed to von Neumann[5], assume a solution in separated variables

$$u_{n,m} = \beta^n \cdot e^{im\phi} \qquad (3.11)$$

Insertion into (2.12) yields

$$\beta = \frac{e^{i\phi}(1+\delta)-1}{e^{i\phi}-(1+\delta)} \qquad (3.12)$$

$|\beta|$ always equals 1 independently of ϕ and δ. This means that the choice of step lengths Δx and Δy will not influence the stability of (2.12). There will be no serious growth of errors, nor will they be reduced.

This simple criterion is confirmed by an ϵ- diagram[6]. Assuming an isolated error ϵ to have occurred in the computation of $u_{n,m}$, the growth of this elementary quantity is illustrated by Figure 3.

m+M	0	$\epsilon(1+M\delta)$	$\epsilon[1+(3M+1)\delta]$	$\epsilon[1+(5M+2)\delta]$		$\epsilon[1+(2MN+M+N)\delta]$
m+2	0	$\epsilon(1+2\delta)$	$\epsilon(1+7\delta)$	$\epsilon(1+12\delta)$		$\epsilon[1+(5N+2)\delta]$
m+1	0	$\epsilon(1+\delta)$	$\epsilon(1+4\delta)$	$\epsilon(1+7\delta)$		$\epsilon[1+(3N+1)\delta]$
m	0	ϵ	$\epsilon(1+\delta)$	$\epsilon(1+2\delta)$		$\epsilon(1+N\delta)$
	0	0	0	0		0
		n	n+1	n+2		n+N

Figure 3. ϵ-diagram for the step-ahead process (2.12).

Disregarding all powers of δ, after N steps in x and M steps in y, a single error will have grown to $\epsilon[1+(2MN+M+N)\delta]$. As $M, N \ll MN$ and $\delta \simeq \frac{\Delta x \, \Delta y}{2}$, the expression for the accumulated error reduces to $\epsilon(1+X \cdot Y)$ where $X = N\Delta x$, $Y = M\Delta y$.

Suppose that an error ϵ has occurred in the first step of the first column. Integrating over, say, a square with side length 3 units in the first quadrant, when the upper right corner (3, 3) of the square is reached, the accumulated error due to the single disturbance will be 10ϵ. Although not prohibitive, this rate of error accumulation is undesirable. Without increasing the truncation error (2.24), the situation can be remedied by using another method of step-ahead integration.

4.1 An analysis of the step-ahead process along the straight line x = (n+1)Δx will show that it is equivalent to the following summation formula:

$$u_{n+1,m+1} = u_{n,m+1} + [\sum_{i=1}^{m+1} (u_{n,i} + u_{n+1,i-1}) + \frac{u_{n+1,0} - u_{n,0}}{\delta}] \delta \qquad (4.11)$$

Here for the first time, the initial values along the characteristic y=0 are entered into the equations by the last term inside the brackets in (4.11).

With the definitions

$$\alpha_{n+1,m} = \sum_{i=1}^{m} (u_{n,i} + u_{n+1,i-1}) + \alpha_{n+1,0} \qquad (4.12)$$

$$\alpha_{n+1,0} = \frac{u_{n+1,0} - u_{n,0}}{\delta} ,$$

a summation step from m to m+1 can be broken up into the following operations:

$$v_{n+1,m+1} = u_{n,m+1} + u_{n+1,m} \qquad (4.14)$$

$$\alpha_{n+1,m+1} = \alpha_{n+1,m} + v_{n+1,m+1} \qquad (4.15)$$

$$u_{n+1,m+1} = u_{n,m+1} + \delta \cdot \alpha_{n+1,m+1} \qquad (4.16)$$

The growth of an isolated error in $u_{n,m}$ during this process is shown in Figure 4, where powers higher than δ^2 have been disregarded.

m+M	0	$\epsilon\delta [1+(M-1)\delta]$	$\epsilon \delta[3+(5M-3)\delta]$	$\epsilon\delta [5+(13M-5)\delta]$		$\epsilon \delta \{2N+1+[M(2N^2+2N+1)-(2N+1)]\delta \}$
m+2	0	$\epsilon \delta (1+\delta)$	$\epsilon \delta (3+7 \delta)$	$\epsilon (5+21 \delta)$		$\epsilon\delta [2N+1+(4N^2+ 2N+1)\delta]$
m+1	0	$\epsilon \delta$	$\epsilon \delta (3+2 \delta)$	$\epsilon (5+8 \delta)$		$\epsilon\delta (2N+1+2N^2 \delta)$
m	0	ϵ	$\epsilon (1+\delta)$	$\epsilon (1+\delta)^2$		$\epsilon (1+\delta)^N$
	0	0	0	0		0
		n	n+1	n+2		n+N

Figure 4. ϵ -diagram for $u_{n,m}$ derived from (4.14)-(4.16).

Here an error propagates mainly along a horizontal line in the diagram. As $N\delta \ll 1$, it will practically not grow. Above the main line of disturbance, the error will propagate according to the expression

$$\epsilon \delta \{2N+1+[M (2N^2+2N+1) - (2N+1)]\delta \}$$

With the same approximations as in section 3.1, this expression reduces to

$$\epsilon\,\delta \cdot N \cdot (XY + 2)$$

For X=Y=3, N=300 this gives 0.165ϵ, which is a considerable improvement compared to the previous process in section 3.1.

The corresponding ϵ-diagram for $\alpha_{n,m}$ shows the same trends, but in limes values are doubled as per the diagram in Figure 3. This is of no consequence, however, as the function wanted is $u_{n,m}$, not $\alpha_{n,m}$.

It should be stressed that the developments of section 4.1 are equally valid if n and m are permuted, i.e., the summations are performed in parallel to the x-axis. The choice is a matter of convenience with respect to the initial functions.

4.2 Integration of equation (1.15) with respect to y for a constant x yields

$$\frac{\partial z(x,y)}{\partial x} = \int_0^y z(x, \beta)\, d\beta + \chi'(x) \qquad (4.21)$$

χ (x) is the initial value of z along the x-axis.

If in (4.11) δ is approximated by $\Delta x \Delta y/2$ and the equation is slightly rewritten, an equivalent of (4.21) in the finite domain will appear

$$\frac{u_{n+1,m+1} - u_{n,m+1}}{\Delta x} = \frac{\Delta y}{2} \left[\sum_{i=1}^{m+1} u_{n,i} + u_{n+1,i-1} \right] + \frac{u_{n+1,0} - u_{n,0}}{\Delta x} \qquad (4.22)$$

The three main terms of (4.22) correspond to the three terms of (4.21).

5.1 Under the usual assumptions of continuity and differentiability the higher derivatives of z with respect to x can be written as

$$\frac{\partial^p z(x,y)}{\partial x^p} = \int_0^y \frac{\partial^{p-1} z(x,\beta)}{\partial x^{p-1}}\, d\beta + \chi^{(p)}(x) \qquad (5.11)$$

Thus every new derivative can be calculated from the preceding one. After introducing the initial values along the y-axis, $z(0,y) = \psi(y)$, where $\psi(0) = \chi(0)$, the solution of (1.15) with the specified initial conditions can be expanded in a Taylor series.

$$z(x,y) = z(0,y) + x \frac{\partial z(0,y)}{\partial x} + \frac{x^2}{\underline{|2}} \frac{\partial^2 z(0,y)}{\partial x^2} + \frac{x^3}{\underline{|3}} \frac{\partial^3 z(0,y)}{\partial x^3} \qquad (5.12)$$

After insertion of (5.11) and observing that

$$\sum_{p=1}^{\infty} \frac{x^p}{\underline{|p}} \chi^{(p)}(0) = \chi(x) - \chi(0), \qquad (5.13)$$

(5.12) yields

$$z(x,y) = \psi(y) + \chi(x) - \chi(0) + \sum_{p=1}^{\infty} \frac{x^p}{\lfloor p} \{ \int_0^y \int_0^{\beta_p} \cdots \int_0^{\beta_2} \psi(\beta_1)\, d\beta_1\, d\beta_2 \ldots d\beta_p + \sum_{q=1}^{p-1} \frac{y^q}{\lfloor q} \chi^{(p-q)}(0) \} \qquad (5.14)$$

For p=1 the multiple integral is to be interpreted as the single integral $\int_0^y \psi(\beta_1)\, d\beta_1$ and the sum $\sum_{q=1}^{p-1}$ as zero.

Within the radius of convergence of the Taylor series, equation (5.14) represents another means of numerical solution of equation (1.15) with the characteristic initial values $\chi(x)$ and $\psi(y)$ along the axes. As x and y are permutable in (1.15), permutation of x and y, χ and ψ in (5.14) will give another resolution formula for z(x, y).

5.2 A series expansion without the restrictions of differentiability on the derivatives of one of the initial functions can be obtained from (1.21), using Picard's substitution method[4]. Assume a first approximation:

$$z_0(x,y) = \psi(y) + \chi(x) - \chi(0) \qquad (5.21)$$

Insertion into (1.21) gives a new approximation

$$z_i(x,y) = z_0 + x \int_0^y \psi(\beta_1)\, \beta_i + y \int_0^x \chi(\alpha_1)\, d\alpha_i - xy \cdot \chi(0) \qquad (5.22)$$

Repetition of this process will give a converging series

$$z(x,y) = \psi(y) + \chi(x) - \chi(0) + \sum_{p=1}^{\infty} \frac{x^p}{\lfloor p} \cdot \int_0^y \int_0^{\beta_p} \cdots \int_0^{\beta_2} \psi(\beta_1)\, d\beta_1\, d\beta_2 \ldots\ldots d\beta_p + \sum_{p=1}^{\infty} \frac{y^p}{\lfloor p} \int_0^x \int_0^{\alpha_p} \cdots \int_0^{\alpha_2} \chi(\alpha_1)\, d\alpha_1\, d\alpha_2 \ldots d\alpha_p - \chi(0) \sum_{p=1}^{\infty} \frac{x^p y^p}{(\lfloor p)^2} \qquad (5.23)$$

This expansion represents the unique solution of (1.15) with the specified initial conditions[4]. The only restriction on $\chi(x)$ and $\psi(y)$ is that they shall have a first derivative throughout the region of integration.

If $\chi(x)$ is supposed to be expansionable in a Taylor series, the identity between (5.23) and (5.14) is easily established. Both these formulas have limited numerical importance because in practice the multiple integrals can seldom be explicitly performed.

6.1 For a test setup on the 604 a specific set of constants and initial values have been chosen. In equations (1.11), (1.12), and (1.13) a and b have been set equal to 1 and c equal to zero.

Therefore, the setup problem is the following*:

$$\frac{\partial^2\phi}{\partial x\,\partial y} + \frac{\partial\phi}{\partial x} + \frac{\partial\phi}{\partial y} = 0 \qquad (6.11)$$

Initial values: For $x = 0$, $\phi(0,y) = 1-e^{-y}$ (6.12)

For $y = 0$, $\phi(x,0) = 0$

By the transformation $\phi = z\,e^{-(x+y)}$ (6.13)

(1.15) will reappear

$$\frac{\partial^2 z}{\partial x\,\partial y} = z \qquad (6.14)$$

with the initial values

For $x = 0$, $z(0,y) = \psi(y) = e^y-1$ (6.15)

For $y = 0$, $z(x,0) = \chi\,(x) = 0$

Insertion of ψ and χ into either of equations (5.14) or (5.23) gives the solution of the problem in the form of the series expansion

$$z(x,y) = e^y-1 + (e^y-1-y)\,x + (e^y-1-y-\frac{y^2}{\lfloor\!\underline{2}})\,\frac{x^2}{\lfloor\!\underline{2}} + \ldots \qquad (6.16)$$

After rearranging the order of the terms, this expression can be written

$$z(x,y) = e^{x+y} - \sum_{p=0}^{\infty} \frac{x^p}{\lfloor\!\underline{p}} \sum_{q=0}^{p} \frac{y^q}{\lfloor\!\underline{q}} \qquad (6.17)$$

and in this form coincides with one of the mentioned Bessel expansions. From (6.13) and (6.16) or (6.17) it is seen that

$$0 \leqq \phi < 1 \qquad (6.18)$$

This limitation on ϕ is going to be used as a check in each computation step. Equation (6.16) is used as a manual check when the machine integration is completed.

6.2 The main computation formula is (4.11), where the term $u_{n+1,0} - u_{n,0}/\delta$ disappears owing to (6.15). Every step in the calculation is checked against equation (2.12).

One run on the 604 represents the integration along a straight line, say, $x = (n+1)\,\Delta x$ in the xy- net, Figure 2. This line will be called column n+1. According

* This case has been treated by C.G. Allander in his doctor's thesis ''Untersuchung des Adsorptionsvorganges in Adsorbentenschichten mit Linearer Adsorptionsisotherme,'' Royal Institute of Technology, Stockholm, 1953. Using the classical Riemann integration method for hyperbolic equations, Allander expresses the solution of equations (6.11), (6.12) as $\phi = e^{-x}\int_0^y e^{-t}\, I_0\,(2\sqrt{xt})\,dt$, where I_0 is the Bessel function of order zero with pure imaginary argument. After expanding the integrand in a series and integrating, he arrives at the expression

$$\phi = 1 - e^{-(x+y)} \sum_0^{\infty} \left(\frac{x}{y}\right)^{\frac{m}{2}} I_m\,(2\sqrt{xy})$$

which is identical with equation (6.17) after multiplication with e^{x+y}.

to (2.12) and (4.11) the values $u_{n+1,m+1}$ are entirely determined by the corresponding values $u_{n,m+1}$ one column back, together with the first value $u_{n+1,0} = 0$ in the new column. Δx and Δy have both been chosen equal to 0.01. This makes δ in (2.13) equal to $0.50001 \cdot 10^{-4}$. This is fed from the digit emitter.

The exponential function is calculated for each integration step with the equation

$$e^{-(x+y)_{n+1,m+1}} = e^{-\Delta y} \cdot e^{-(x+y)_{n+1,m}}, \tag{6.21}$$

where $e^{-\Delta y} = 0.9900498$. It is fed from the digit emitter as 0.99005.

$e^{-(x+y)_{n+1,m}}$ is available from the previous step. Only the first value of a new column $e^{-(x+y)_{n+1,0}}$ has to be read from the card which is used for starting the integration process.

7.1 In the test runs the integration has been carried out to 3 units in y. This means that there are 300 cards for the values $u_{n,m}$. Before each run these master cards are interspersed with blank cards, which are going to be punched in columns 18 - 25 with the results $u_{n+1,m}$ of the actual run and then become the master cards of the next integration run n+2.

The blanks are also punched in columns 27 - 33 with $\phi_{n+1,m}$, which is the ultimate information wanted, and in columns 34 - 41 with $e^{-(x+y)_{n+1,m}}$, which is punched out only to be read into the machine again from the second reading station and used according to (6.21). In this respect the card is used as a moving auxiliary storage, to remedy the limited storage capacity of the 604.

The mentioned quantities are punched from the counter and general storage of the 604 as specified in the flow diagram, Figure 5, below the heading "1st Card Cycle." The blank card is also punched with the number m for identification in columns 5 - 7 and with $u_{n,m}$ in columns 9 - 16. Both these quantities are read from the preceding master card m in the second reading station. $u_{n,m}$ is also to be read into the machine again during the second card cycle when the former blank card reaches the second reading station for use in the check formula (2.12). The electronic storage does not suffice for keeping this quantity over the two machine cycles which together constitute one computing step.

The only original information in the blank cards is the x-coordinate number n+1 which is gang-punched into columns 1 - 3 before the merging with master cards. The master cards are not punched during the actual run, only during the preceding run when they were blanks. They control various machine functions with an X-punch in column 80. The blank cards have to be punched with this X in order to be distinguished as masters in the next integration run.

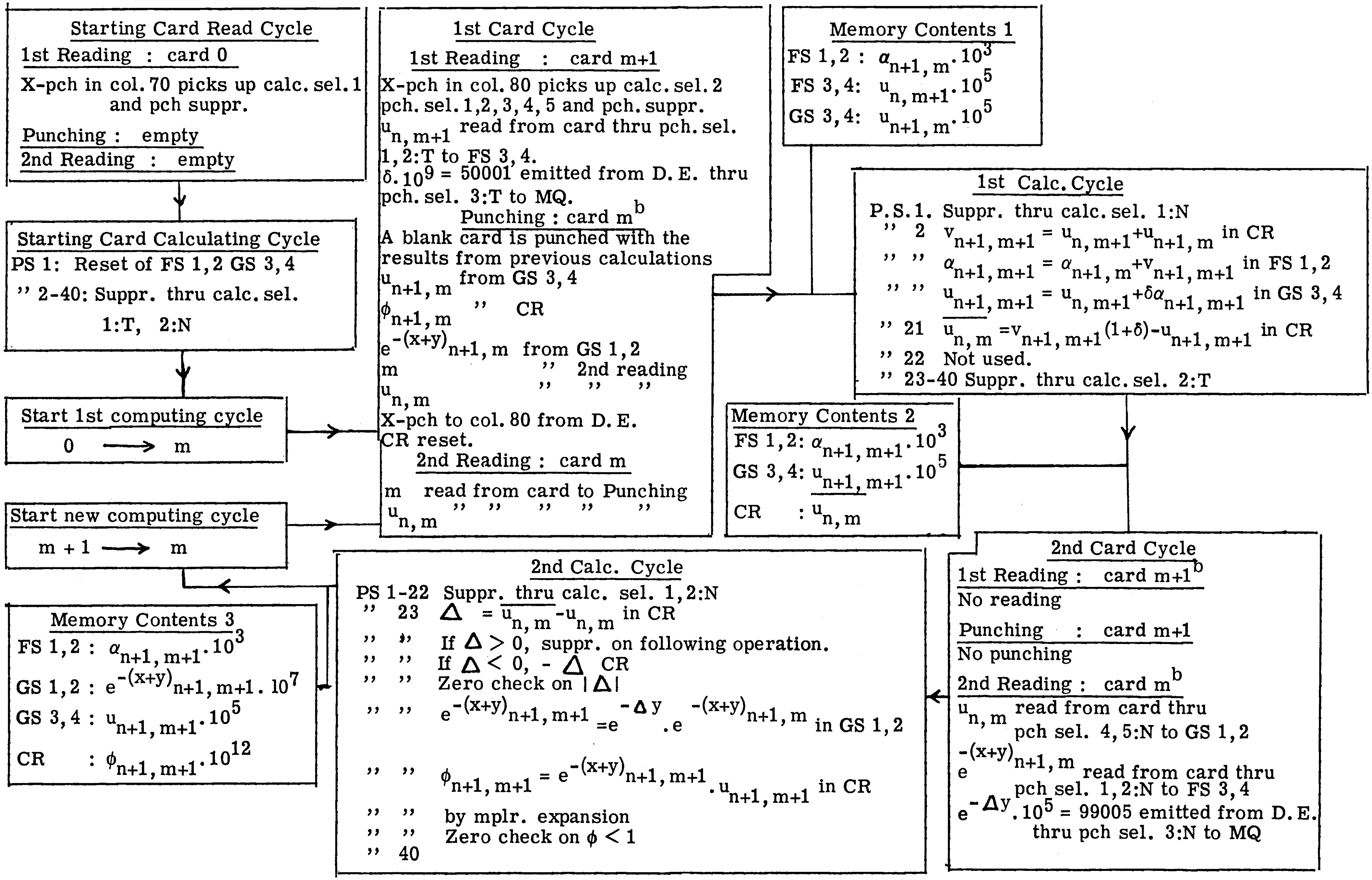

Starting Card Read Cycle
1st Reading : card 0
X-pch in col. 70 picks up calc. sel. 1 and pch suppr.
Punching : empty
2nd Reading : empty
Starting Card Calculating Cycle
PS 1: Reset of FS 1,2 GS 3,4
" 2-40: Suppr. thru calc. sel. 1:T, 2:N
Start 1st computing cycle
0 ⟶ m
Start new computing cycle
m + 1 ⟶ m
Memory Contents 3
FS 1,2 : $\alpha_{n+1,m+1} \cdot 10^3$
GS 1,2 : $e^{-(x+y)}{}_{n+1,m+1} \cdot 10^7$
GS 3,4 : $u_{n+1,m+1} \cdot 10^5$
CR : $\phi_{n+1,m+1} \cdot 10^{12}$
1st Card Cycle
1st Reading : card m+1
X-pch in col. 80 picks up calc. sel. 2 pch. sel. 1,2,3,4,5 and pch. suppr.
$u_{n,m+1}$ read from card thru pch. sel. 1,2:T to FS 3,4.
$\delta \cdot 10^9 = 50001$ emitted from D. E. thru pch. sel. 3:T to MQ.
Punching : card m^b
A blank card is punched with the results from previous calculations
$u_{n+1,m}$ from GS 3,4
$\phi_{n+1,m}$ " CR
$e^{-(x+y)}{}_{n+1,m}$ from GS 1,2
m " 2nd reading
$u_{n,m}$ " " "
X-pch to col. 80 from D. E.
CR reset.
2nd Reading : card m
m read from card to Punching
$u_{n,m}$ " " " " "
Memory Contents 1
FS 1,2 : $\alpha_{n+1,m} \cdot 10^3$
FS 3,4: $u_{n,m+1} \cdot 10^5$
GS 3,4: $u_{n+1,m} \cdot 10^5$
1st Calc. Cycle
P.S. 1. Suppr. thru calc. sel. 1:N
" 2 $v_{n+1,m+1} = u_{n,m+1} + u_{n+1,m}$ in CR
" " $\alpha_{n+1,m+1} = \alpha_{n+1,m} + v_{n+1,m+1}$ in FS 1,2
" " $u_{n+1,m+1} = u_{n,m+1} + \delta\alpha_{n+1,m+1}$ in GS 3,4
" 21 $\overline{u}_{n,m} = v_{n+1,m+1}(1+\delta) - u_{n+1,m+1}$ in CR
" 22 Not used.
" 23-40 Suppr. thru calc. sel. 2:T
Memory Contents 2
FS 1,2: $\alpha_{n+1,m+1} \cdot 10^3$
GS 3,4: $u_{n+1,m+1} \cdot 10^5$
CR : $u_{n,m}$
2nd Calc. Cycle
PS 1-22 Suppr. thru calc. sel. 1,2:N
" 23 $\Delta = \overline{u}_{n,m} - u_{n,m}$ in CR
" " If $\Delta > 0$, suppr. on following operation.
" " If $\Delta < 0$, $-\Delta$ CR
" " Zero check on $|\Delta|$
" " $e^{-(x+y)}{}_{n+1,m+1} = e^{-\Delta y} \cdot e^{-(x+y)}{}_{n+1,m}$ in GS 1,2
" " $\phi_{n+1,m+1} = e^{-(x+y)}{}_{n+1,m+1} \cdot u_{n+1,m+1}$ in CR
" " by mplr. expansion
" " Zero check on $\phi < 1$
" 40
2nd Card Cycle
1st Reading : card $m+1^b$
No reading
Punching : card m+1
No punching
2nd Reading : card m^b
$u_{n,m}$ read from card thru pch sel. 4,5:N to GS 1,2
$e^{-(x+y)}{}_{n+1,m}$ read from card thru pch sel. 1,2:N to FS 3,4
$e^{-\Delta y} \cdot 10^5 = 99005$ emitted from D. E. thru pch sel. 3:N to MQ

Fig. 5 Flow Diagram of Type 604 Operations

7.2 The processing of information in position and time is illustrated in Figure 5, Flow Diagram of Type 604 Operations, and in Figure 6, Card Diagram of Type 604 Operations.

The punching of a blank card m^b, described abov^e, occurs in the beginning of a new computing step. The results of the previous computing step are punched out simultaneously with the reading of new data from master card m+1 for use in the first of the two calculating cycles in the new computing step. $u_{n,m+1}$ is read through punch selectors 1,2, transferred by the X-punch in column 80 of the master card, into factor storage 3,4, assigned together. The decimal point convention is such that $u_{n,m+1}$ will have 5 decimals and 3 integer digits.

The constant $\delta = 50001.10^9$ is sent from the digit emitter to the multiplier quotient unit through punch selector 3, which is also controlled by the X-punch in column 80.

Selectors must be used for reading into various storages because in the second card cycle other data have to be read from other sources into the same units.

The symbolic Memory Contents 1 in the flow diagram, Figure 5, indicates that in addition to $u_{n,m+1}$ other information is stored since the previous calculating step. $u_{n+1,m}$ is stored in general storage 3,4 with the same decimal convention as for $u_{n,m+1}$. The quantity $\alpha_{n+1,m}$, defined in (4.12), is stored with 3 decimals and 5 integers in factor storage 1,2.

7.3 The operations during the first calculating cycle are shown in the flow diagram. The calculation is performed according to formulas (4.14) - (4.16), leaving the new values of $\alpha_{n+1,m+1}$ and $u_{n+1,m+1}$ in the same storages as the old values, shown by Memory Contents 2 in the flow diagram. In order to economize with storage, the check formula (2.12) is rearranged to

$$\overline{u_{n,m}} = v_{n+1,m+1} \ (1+\delta) - u_{n+1,m+1} \ , \qquad (7.31)$$

where the bar over $u_{n,m}$ indicates that it represents a back solution from the newly formed values of $u_{n+1,m+1}$ and $v_{n+1,m+1}$.

$\overline{u_{n,m}}$ is left in the counter after the first calculating cycle. The operations described are performed on program steps 2-22. Unlike the remaining program steps they are not suppressed, because calculator selector 2 is picked up by the X-punch in column 80.

During the second card cycle the master card m+1 is moved to the punching station. No punching occurs, as the punch suppression was activated during the previous card cycle. A new blank card $m+1^b$ has entered the first reading station without being read. The previous blank card m^b, which is now carrying information, is read from the second reading station, as shown in Figure 5. This results in the transfer

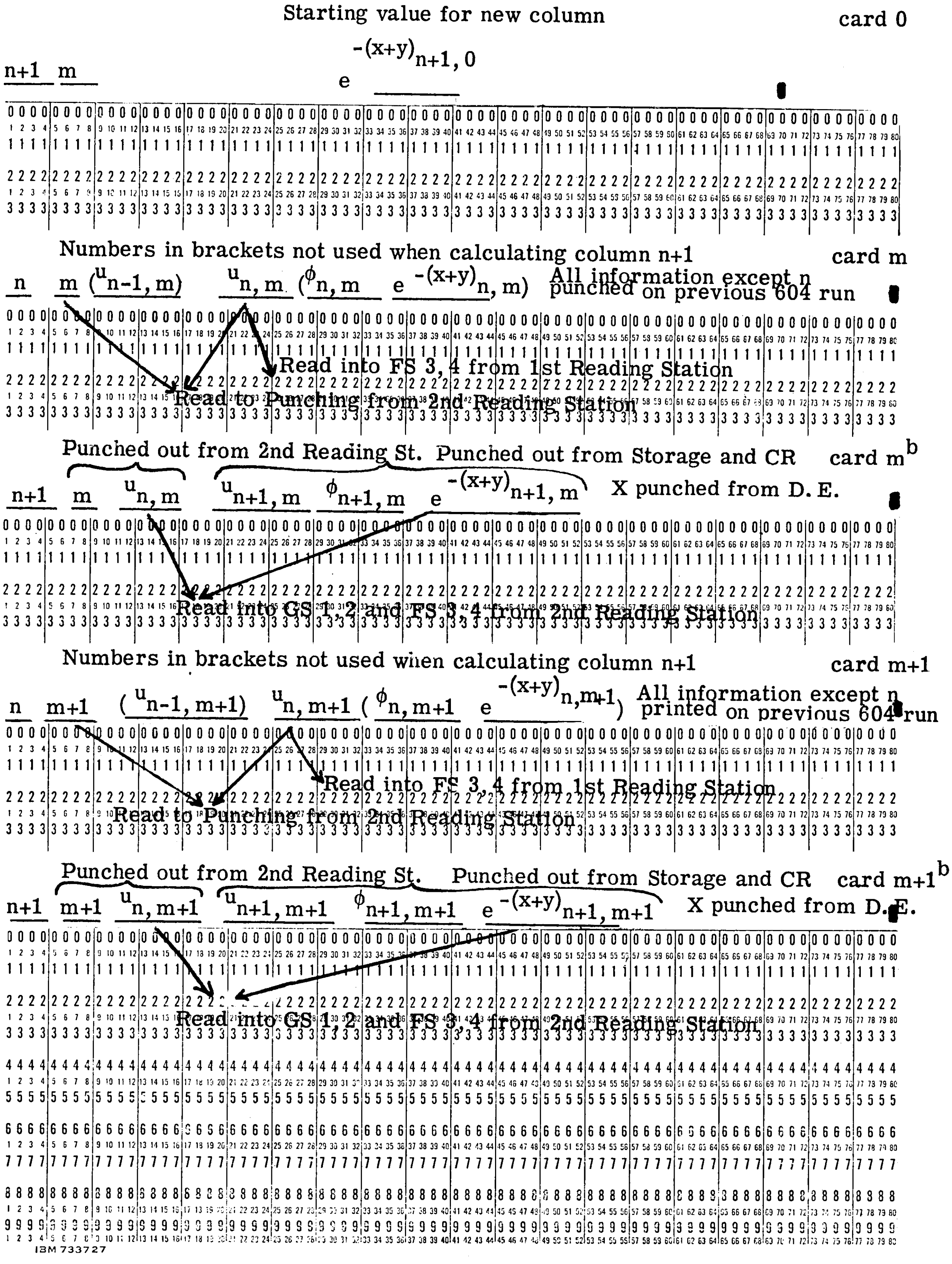

Starting value for new column
card 0
n+1
m
e^{-(x+y)}n+1, 0
Numbers in brackets not used when calculating column n+1
card m
n
m
(un-1, m)
un, m
(φn, m
e^{-(x+y)}n, m)
All information except n punched on previous 604 run
Read into FS 3, 4 from 1st Reading Station
Read to Punching from 2nd Reading Station
Punched out from 2nd Reading St.
Punched out from Storage and CR
card m^b
n+1
m
un, m
un+1, m
φn+1, m
e^{-(x+y)}n+1, m
X punched from D. E.
Read into GS 1, 2 and FS 3, 4 from 2nd Reading Station
Numbers in brackets not used when calculating column n+1
card m+1
n
m+1
(un-1, m+1)
un, m+1
(φn, m+1
e^{-(x+y)}n, m+1)
All information except n printed on previous 604 run
Read into FS 3, 4 from 1st Reading Station
Read to Punching from 2nd Reading Station
Punched out from 2nd Reading St.
Punched out from Storage and CR
card m+1^b
n+1
m+1
un, m+1
un+1, m+1
φn+1, m+1
e^{-(x+y)}n+1, m+1
X punched from D. E.
Read into GS 1, 2 and FS 3, 4 from 2nd Reading Station
IBM 733727

Fig. 6 Card Diagram of Type 604 Operations

of the quantities $u_{n,m}$ and $e^{-(x+y)}{}_{n+1,m}$ from the card to general storage 1,2 and factor storage 3,4. The transfer goes through the normal side of punch selectors 1,2, 4,5, because there is no X-punch in the blank card $m+1^b$ in the first reading station.

During the second calculating cycle, program steps 1 - 22 are suppressed through the normal side of calculator selectors 1,2 (see the calculate control panel wiring, Figure 7). The operations performed on program steps 23 - 40 are shown in the flow diagram.

A check quantity $\Delta = \overline{u_{n,m}} - u_{n,m}$ is the difference between the back solution stored in the counter since the first calculating cycle and the original value of $u_{n,m}$. The two values shall only differ by some units in the last position due to rounding errors. As the difference can be either positive or negative, its absolute value $|\Delta|$ is entered into the counter by reading Δ out to general storage 1,2 and subtracting it into the counter if it is negative. If it is positive, the subtraction is suppressed. The absolute value of the difference is tested for zeroes in the second position and upwards. These operations are performed in program steps 23-26.

The rest of the second calculating cycle is used for computation of a new value of $e^{-(x+y)}{}_{n+1,m+1}$ according to (6.21) and of $\phi_{n+1,m+1}$ according to (6.13). During the second card cycle, $e^{-\Delta y} \cdot 10^5 = 99005$ was read into the multiplier quotient unit from the digit emitter through the normal side of punch selector 3. Calculating the new value of $e^{-(x+y)}{}_{n+1,m+1}$ takes just one multiplication. When this 8-digit number is multiplied by $u_{n+1,m+1}$ to give ϕ, it has to be split into two parts, called e_1 and e_2. The products ϕ_1 and ϕ_2 of e_1 and e_2,with $u_{n+1,m+1}$ appropriately shifted, are added together on program step 39 to make $\phi_{n+1,m+1}$.

When shifting out the three most significant digits of the product $\phi_1 \cdot 10^7$, the validity of (6.18), i.e., $\phi < 1$, is checked on program step 36 by a zero test on the digits to the left of the decimal point. $\phi_{n+1,m+1}$ is punched from the counter, $e^{-(x+y)}{}_{n+1,m+1}$ from general storage 1,2,and $u_{n+1,m+1}$ from general storage 3,4. The punching occurs during card cycle 1, which belongs to the next computation step. This means that the loop in the flow diagram has been closed, which is indicated by the symbol $m+1 \rightarrow m$.

7.4 Thus the calculation proceeds until all the cards in the feed hopper are finished. After the 604 run, the old master cards are sorted away for tabulation while the new master cards, i.e., the previous blanks, are interspersed with new blanks. The merged deck is preceded by a starting card, which carries an X-punch in column 70 and the starting value of $e^{-(x+y)}{}_{n+2,0}$ in columns 34-41. The role of the starting card, called card 0, is shown in the flow diagram. On the first reading cycle, the X-punch picks up calculator selector 1 and punch suppression. In the following calculating cycle, program step 1 is the only one which is not suppressed, due to the wiring of calculator selectors 1 and 2, as shown in the calculator control panel wiring diagram, Figure 7. Program step 1 resets $\alpha_{n+2,0}$ and $u_{n+2,0}$ to zero in factor storage 1,2 and general storage 3,4. Then everything is ready for the first computing step of the new integration column.

The starting card is not punched because of the punch suppression, but it is treated as a former blank when reaching the second reading station. As nothing has been key-punched in the columns corresponding to $u_{n+1,0}$, this quantity is correctly set as zero when read into general storage 1,2. The starting value of the exponential function, $e^{-(x+y)}{}_{n+2,0}$ is read into factor storage 3,4.

7.5 From the card diagram, Figure 6, it is evident that several of the fields are used only when the card is run through the machine for the first time, not the second time when it is a master card. This means that the starting deck, representing the values on the y-axis, does not need the quantities enclosed in brackets on the master cards. The only information necessary to key-punch is $n = 0$, m and $\psi(y) = u_{0,m} = e^{m\,\Delta y}-1$. The starting deck for the first column is to be distinguished from the starting cards of which one has to precede every merged deck of masters and blanks.

7.6 The programmed checks insure a completely error-free computation and punching of $u_{n,m}$ under the condition that the original values of $u_{0,m} = \psi(y)$ are correct, which can be checked once and for all. The computation of $e^{-(x+y)}$ is easily checked on the tabulator by totalling

$$S_1 = \sum_{m=1}^{M} e^{-(x+y)}{}_{n,m}$$

on the same run on which the results are listed. For instance, a mistake in choosing the starting card might not influence the computation of $u_{n,m}$. The error in $e^{-(x+y)}{}_{n,m}$ immediately shows up when checking the total.

For $M=300$, $S_1 e^{n\Delta x}$ has shown to deviate from the exact value $1-e^{-3}/e^{0.01}-1$ by an amount which is almost entirely due to the approximation of $e^{-0.01}$ with 5 figures in the 604 setup.

The totals of $\sum_{m=1}^{M} u_{n,m}$ and $\sum_{m=1}^{M} u_{n+1,m}$ are listed in the same tabulation run. Because every such total is listed twice, this will render a further, although not necessary, check of the punching of $u_{n,m}$. The final calculation of ϕ according to (6.13) is unchecked due to the shortage of program steps. On a 60-program step machine a check could be easily made part of the program.

8.1 The decimal point convention chosen for u and α sets the limits $u < 1000$, $\alpha < 100000$. This would do for integration up to (3.7, 3.7) in the xy-plane.

The test integration was made only to $y = 3$ and $x = 0.43$. A check on the accuracy of the results was obtained through the expansion formula (6.16). Some of the

IBM TYPE 604 ELECTRONIC CALCULATING PUNCH CONTROL PANELS IBM

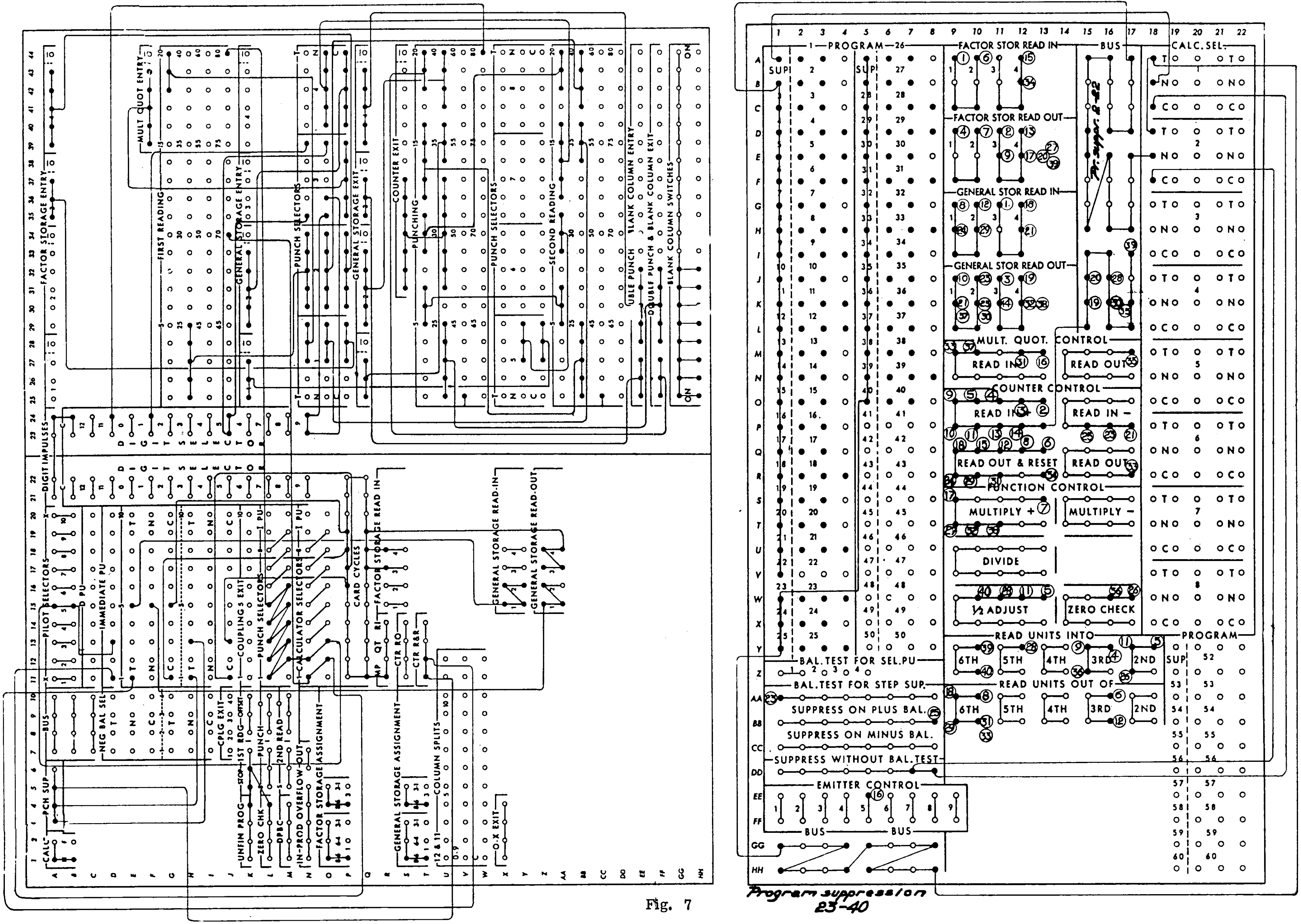


Fig. 7

Program suppression 23-40

values are shown in the table below.

y = 3.00 x	Calc. value $u_{n,m}$	Exact value z(x,y)	Difference $(u_{n,m}-z) \cdot 10^5$
0.01	19.24698	19.24697	1
0.06	20.07180	20.07178	2
0.11	20.92666	20.92664	2
0.16	21.81250	21.81246	4
0.21	22.73025	22.73021	4
0.26	23.68090	23.68085	5
0.31	24.66545	24.66539	6
0.36	25.68493	25.68486	7
0.41	26.74039	26.74030	9

Table 1

The accuracy of the calculated exponential function is shown in Table 2 for the x-column n=25. The starting value $e^{-n\ \Delta x} = e^{-0.25} = 0.7788008$.

x = 0.25 m	Calculated value $e^{-n\ \Delta x}.e^{-m\Delta y}$	Exact value $e^{-(x+y)}$	Difference $\Delta.10^7$	Relative Difference $\Delta_r .10^6$
0	0.7788008	0.7788008		
50	0.4723706	0.4723666	40	8
100	0.2865098	0.2865048	50	18
150	0.1737787	0.1737739	48	28
200	0.1054027	0.1053992	35	33
250	0.0639304	0.0639280	24	38
300	0.0387756	0.0387740	16	41

Table 2

8.2 Although in the test setup the initial function $\chi(x)$ along the x-axis has been chosen as zero, only a minor change is necessary to run the same setup with any regular function $\chi(x)$.

In the starting card for a new column need only be key-punched the values

$$u_{n+1,0} \equiv \chi(\ (n+1)\ \Delta\ x)$$

and

$$\alpha_{n+1,0} \equiv \frac{u_{n+1,0} - u_{n,0}}{\delta} \simeq \frac{2}{\Delta y}\ \ \chi'(\ (n+1/2)\ \Delta x\ )$$

These values are read into general storage 3,4 and factor storage 1,2 on the starting card read cycle. The starting card calculating cycle is skipped.

Consider the general case of equation (1.11). The only difference from the described setup is that an exponential function $e^{-bKx/(ab-c)-ay/K}$ has to be calculated with an equivalent of (6.21) by multiplying with the constant $e^{-a\ \Delta y}/\ K$ in each step.

This means only that other digits are read from the digit emitter into the multiplier unit during the second card cycle and that some readjustments of the number of shift steps in the following operations might be necessary. The decimal point convention for $u_{n,m}$ and $\alpha_{n,m}$ may also have to be changed.

The calculation of the initial values of $\chi(x)$ and $\psi(y)$ from the corresponding values of ϕ is made once and for all before key-punching the starting decks and does not concern the machine setup.

If it is accepted that the wanted quantity ϕ is given as a function of x and y instead of its original independent variables ξ and η, it has been demonstrated how the canonical homogeneous case of the linear hyperbolic equation with constant coefficients and characteristic initial values can be set up for numerical integration on the 604 Electronic Calculating Punch.

References

1. Goursat, E. Cours d'Analyse Mathématique, Tome III, 5:e éd., Gauthier-Villars, Paris (1942), Article 479.
2. Ibid., Article 499.
3. Ibid., Article 489.
4. Ibid., Article 494.
5. Hildebrand, F. B. Methods of Applied Mathematics, Prentice Hall, New York (1952), pp. 339 - 340.
6. Collatz, L. Numerische Behandlung von Differentialgleichungen (Die Grundlehren der mathematischen Wissenschaften Band LX), Springer-Verlag, Berlin, Göttingen, Heidelberg (1951), p. 205.
7. Bateman, H. Partial Differential Equations of Mathematical Physics, Cambridge University Press, London (1932), Sections 2.15, 2.16.

AN APPROACH TO STORED PROGRAMMING ON THE 607 ELECTRONIC CALCULATOR

Eric V. Hankam

Watson Scientific Computing Laboratory
International Business Machines Corporation

1. Introduction

The general purpose computing system described in the following paragraphs has been designed to make maximum use of the high-speed program unit and electronic storage of the 607. In this system as many as six three-address instructions can be read from any given card into the 607 storage. The 607 then interprets and executes these instructions sequentially before the next card is fed. Arithmetical, logical, and shift operations can be performed on 8-digit numbers (fixed decimal) with sign. It is possible to execute alternative instructions, skip instructions, and modify instructions, depending on the result of a logical operation. For convenient use of this system, a simple coding scheme has been developed. Sections 2 to 5 contain sufficient information to enable a coder to apply the system to a problem; sections 6 to 10 are directed toward the reader who is interested in the construction of the control panels.

2. Data

The data used in a calculation are 8-digit numbers with sign. Up to ten 8-digit numbers may be stored in the machine; the ten storage locations are identified by the addresses 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. A number read out of storage is retained in storage, and it may be read out again if required. A number read into storage will replace the number previously occupying that storage location; i.e., a storage location is cleared automatically prior to entry of a new number.

Input: Data may be read from cards into storage locations 0, 1, 2, 3.
Output: Results may be punched into cards from storage locations 8, 9.

Provision has been made to read additional data into storage locations 4, 5, 6 from a special data card (see sec. 5b). To permit independent entry of data, card reading is under the control of X-punches. Similarly, punching is governed by individual X-punches in the card fields associated with storage locations 8 and 9 (see sec. 5a). It should be noted that a number that is punched from storage will also be retained in storage.

3. Instructions

An instruction (A⊙B=C) is a 5-digit number consisting of three single-digit addresses (A, B, C) and a 2-digit operation code ⊙. The meaning of the A-, B-, and C-addresses can be best explained by an illustration. For example, the instruction 1⊙08 (here, ⊙ is an arithmetical operation) will cause the machine to search for the contents of storage locations 1 and 0, to perform an operation on them, and to read the result into storage location 8. In another example, where ⊙ represents multiplication, the instruction 3⊙33 will replace the number that is in storage location 3 by its square.

Five instructions may be read from a single card. These instructions, which are punched into five consecutive card fields, are loaded into 607 storage and executed sequentially, i.e., from left to right as they appear on the card. The number of instructions that will be executed from a given card depends on the 2-digit number, n, punched in that card. For n=00 (or blank), no instruction is executed. The first instruction is always destroyed in the process of execution, but the remaining instructions are circulated in the machine so that, for n >5, instructions 2, 3, 4, 5 are repeated in a "loop." A sixth instruction may be included in that loop as explained in the next paragraph. After the last instruction has been executed, the next card feeds.

Input and output data may be punched into the card that contains the instructions (see card form in sec. 5a). However, an instruction may apply to data already in storage as well as to the data read from this card. It is possible to punch a sixth instruction into the card field normally reserved for input data associated with storage location 0.

4. Operations

a. List of operations and codes

Subtract	A-B	00	allow for carry	
		01	do not allow for carry	
Add	A+B	10	allow for carry	
		11	do not allow for carry	
Multiply	$A \cdot B$	20	allow for 16-digit product	
		21	allow for 15-digit product	
Divide	A/B	30	allow for $\lvert B \rvert \leqq \lvert A \rvert < 10 \lvert B \rvert$	
		31	allow for $\lvert B \rvert > \lvert A \rvert$	
Shift Left	A	40	1 position	
		41	2 positions	
Balance Test	A	50	alternative instructions	
		51	modify instructions	
Zero Test	A	60	A=0	skip instructions
		61	$\lvert A \rvert < 10$	

b. Detailed description of operations

Subtract A-B⟶C, code 00

The number in the storage location specified by the B-address is subtracted from the number in the storage location specified by the A-address; allowance is made for a carry, the right-most digit is dropped, and the result is read into the storage location specified by the C-address.

$$\begin{array}{rl} \mathrm{xxxxxxxx} & A \\ \mathrm{xxxxxxxx} & B \\ \hline \underbrace{\mathrm{xxxxxxxx}}_{C}|\mathrm{x} & \end{array}$$

Subtract A-B⟶C, code 01

The number in the storage location specified by the B-address is subtracted from the number in the storage location specified by the A-address; no allowance is made for a carry, and the result is read into the storage location specified by the C-address.

$$\begin{array}{rl} \mathrm{xxxxxxxx} & A \\ \mathrm{xxxxxxxx} & B \\ \hline \mathrm{xxxxxxxx} & C \end{array}$$

Add A+B⟶C, code 10

Similar to operation, 00.

$$\begin{array}{rl} \mathrm{xxxxxxxx} & A \\ \mathrm{xxxxxxxx} & B \\ \hline \underbrace{\mathrm{xxxxxxxx}}_{C}|\mathrm{x} & \end{array}$$

Add A+B⟶C, code 11

Similar to operation, 01.

$$\begin{array}{rl} \mathrm{xxxxxxxx} & A \\ \mathrm{xxxxxxxx} & B \\ \hline \mathrm{xxxxxxxx} & C \end{array}$$

Multiply A·B⟶C, code 20

Multiply A by B, allow for a 16-digit product, round and drop 8 digits, and store product in C.

$$\underset{A}{(\mathrm{xxxxxxxx})}\;\underset{B}{(\mathrm{xxxxxxxx})} = \underbrace{\mathrm{xxxxxxxx}}_{C}\Big|\underset{5\nwarrow\text{round}}{\mathrm{xxxxxxxx}}$$

Multiply A·B→C, code 21

Multiply A by B, allow for a 15-digit product, round and drop 7 digits, and store product in C.

$$\underset{A}{(xxxxxxxx)}\ \underset{B}{(xxxxxxxx)} = 0\underbrace{xxxxxxxx}_{C}\Big|\underset{5\nwarrow\text{round}}{xxxxxxx}$$

Divide A/B→C, code 30

Divide A by B, allow for 9 quotient digits, round and drop rightmost digit, and store quotient in C.

If $|B| \leqq |A| < 10\,|B|$, the quotient will have 8 significant digits.

If $|B| > |A|$, the quotient will have <8 significant digits.

This operation will cause a quotient overflow and result in a zero quotient for all cases $|A| \geqq 10\ \ |B|$

$$\begin{matrix}A\\B\end{matrix}\frac{xxxxxxxx\cdot 10^{8}}{xxxxxxxx} = \underbrace{xxxxxxxx}_{C}\Big|\underset{5\nwarrow\text{round}}{x}$$

Divide A/B→C, code 31

Divide A by B, allow for 9 quotient digits, round and drop rightmost digit, and store quotient in C.

If $|A| < B \leqq 10\,|A|$, the quotient will have 8 significant digits.

If $|B| > 10\ |A|$, the quotient will have <8 significant digits.

This operation will cause a quotient overflow and result in a zero quotient for all cases $|A| \geqq |B|$.

$$\begin{matrix}A\\B\end{matrix}\frac{xxxxxxxx\cdot 10^{9}}{xxxxxxxx} = 0\underbrace{xxxxxxxx}_{C}\Big|\underset{5\nwarrow\text{round}}{x}$$

Shift Left A→C, code 40

A is shifted one place to the left and the result is stored in C. Using a C-address equal to the A-address has the effect of returning the shifted number to its original storage location. The B-address has no function in this operation and can be left blank.

Shift Left A→C, code 41

A is shifted two places to the left, and the result is stored in C. As in operation 40, the B-address has no function in this operation and can be left blank.

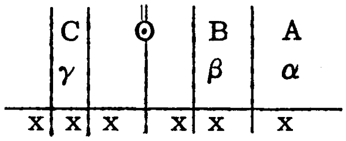
Balance Test A, code 50

The algebraic sign of A is tested. If A is negative, instructions 1 and 2 will be ignored in subsequent cards until a card that has a Y-punch in column 5 is read; instructions 1 and 2 in the Y_5-card will also be ignored, but, thereafter, they will be executed in the normal manner. For example, when n=2, this "branching" operation will cause instructions 1 and 2 to be executed if A is positive, or instructions 3 and 4 if A is negative.

As a by-product of the design of the control panel, B will transfer into C. In general, both the B-address and C-address will be left blank.

Balance Test A, code 51

The algebraic sign of A is tested. If A is negative, the A-, B-, and C-addresses of instructions in all subsequent cards will be modified by α , β , and γ, respectively, until a card with a Y-punch in column 6 is read. Instructions in the Y_6-cards will also be modified, but, thereafter, they will be executed in the normal manner. The single digit numbers, α , β, and γ, are punched in the card and are applied additively to the addresses of an instruction to transform them to $A+\alpha$, $B+\beta$, and $C+\gamma$. The section on the 607 program shows that the addition is performed as follows:



A carry resulting from $(A+\alpha)$ will increase $(B+\beta)$; likewise ⊙ may be increased by 1. If $C + \gamma > 9$, it will be treated as 9. By means of a single X-punch, minus signs may be attached to α, β , and γ . Though α , β , γ may change from card to card, all instructions in a single card will be modified by the same α , β , γ .

As a by-product of the design of the control panel, B will transfer into C. In general, both the B-address and C-address will be left blank.

Zero Test A, code 60

A is tested for zero. If A is zero, all subsequent instructions in the same and following cards are skipped until a card with a Y-punch in column 7 is read. Instructions in the Y_7-card will also be skipped, but, thereafter, they will be executed in the normal manner.

As a by-product of the design of the control panel, B will transfer into C. In general, both the B-address and the C-address will be left blank.

Zero Test A, code 61

The only difference between this operation and operation 60 is that all positions except the rightmost position of A are tested for zero, i.e., operation 61 tests whether $|A| < 10$.

c. Further remarks on operations.

A 7-, 8- or 9-punch in the tens position of the operation code will be treated as "No Operation." In this operation the A-address has no function, but B may be transferred into C. A number may also be transferred from one storage location to another by the addition of zero to it, but the former method has the advantage of not requiring a storage location for the number zero.

The sign of a number can be changed by the subtraction of the number from zero.

A number may be shifted to the right, one place at a time, by the addition of zero to it by means of operation 10. Since the low-order positions of a product are dropped, we may shift a number k places to the right by multiplying it by 10^{8-k} (where $k = 1, 2, \ldots, 8$) with operation 20. In the latter case, rounding will take place if the right shift causes significant digits to be dropped.

Since the 607 reads blank columns as zeros, an unpunched position in an instruction will always be treated as a zero. In particular, the machine will consider a completely blank instruction as 0 00 0 0, and if it executes such an instruction (subject to n), it will, in effect, clear storage location 0. As a matter of fact, any storage location can be cleared by specification of its address as the C-address in an otherwise blank instruction.

A study of the section describing the 607 program will show that the machine simply tests whether the units position of the operation code is zero or not; therefore, the 1 in the units position is really arbitrary, and any other non-zero digit can be substituted.

5. Card Form

a. General card form for instructions and data

identification	n	YYY $\alpha\beta\gamma$	instr #1	instr #2	instr #3	instr #4	instr #5	XX data into storage 0	X data into storage 1	X data into storage 2	X data into storage 3	X punch from storage 8	X punch from storage 9
1-2	3-4	5-7	8-12	13-17	18-22	23-27	28-32	33-40	41-48	49-56	57-64	65-72	73-80

Col. 1,2 : identification

Col. 3,4 : n (number of instructions to be executed)

Col. 5-7 : α , β , γ , (address modifiers). For a complete explanation of α , β , γ , and the significance of the Y-punches see the description of the Balance Test and Zero Test operation (sec. 4b). An X-punch in col. 7 will make α , β , γ negative.

Col. 8-32 : five 5-position instruction fields. The leftmost position of each field contains the A-address; the 2nd and 3rd positions contain the operation code Θ; the 4th position contains the B-address; and the last position contains the C-address.

Col. 33-64 : four 8-position input-data fields are used to enter data into storage locations 0, 1,2,3, respectively. An X-punch in the leftmost position of each field will cause the corresponding storage location to read in. An X-punch in the rightmost position of the field identifies negative numbers.
An X-punch in col. 35 makes it possible to use col. 36-40 for a sixth instruction. An X-punch in col. 33 is also required to read the instruction into storage location 0.

Col. 65-80 : two 8-position output-data fields are used to punch results from storage locations 8,9, respectively. An X-punch in the leftmost position of each field will cause the corresponding storage location to read out. An X-punch in the rightmost position of the field identifies negative numbers.

b. Special data card form (Y in col. 1)

Y	X data into storage 4	X data into storage 5	X data into storage 6	
1	9-16	17-24	25-32	

A Y-punch in col. 1 makes it possible to read data from three adjacent 8-position fields (col. 9-32) into storage locations 4, 5, 6, respectively. An X-punch in the leftmost position of each field will cause the corresponding storage location to read in. An X-punch in the rightmost position of the field identifies negative numbers.

Instructions cannot be read from this card, but instructions already in storage will not be destroyed, and can be executed if n $\neq$ 0 is punched into col. 3, 4.

c. Further remarks on card forms.

Card forms are quite arbitrary, and we can alter them readily by making corresponding changes on the 529 control panel. Suggestions for some modifications appear in the Appendix.

6. <u>607 Storage</u>

a. Data Storage

The following 607 storage units have been used for storage locations 0, 1, . . . , 9:

0 ~	G1, 2
1 ~	G3, 4
2 ~	G9, 10
3 ~	G11, 12
4 ~	G13, 14
5 ~	G15, 16
6 ~	G17, 18
7 ~	G19, 20
8 ~	G21, 22
9 ~	G23, 24

The 607 at the Watson Scientific Computing Laboratory is a model with 140 program steps and, therefore, does not have G5, 6, 7, 8.

b. Working Storage

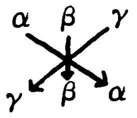
F1, 2	stores A
F3, 4	stores B before an operation and C after an operation
MQ	
EC	

c. Miscellaneous Storage

F5 stores n — Two card columns have been reserved for n. If a third card column were included on the 529 control panel, the 607 program could handle any

F7 stores $\alpha\beta\gamma$ — $n \leqq 999$. The 529 control panel has been wired so that α and γ are interchanged as they are read from the card into F7, i.e.,

on card

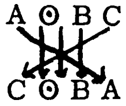


in F7

d. Instruction storage

F6	instruction	1
F8	”	2
F10	”	3
F12	”	4
F9, 11	”	5

The 529 control panel has been wired so that every instruction is read from the card into storage as follows:



on card

in storage

G1, 2 may also be used for instruction 6.

7. Explanation of the 607 program

In the following, the ith program step will be denoted by P_i.

a. P_1 to P_3 and P_{10}: n is reduced by 1 and tested. If $n \geqq 0$, program repeat is impulsed; if $n < 0$, P_{10} will be active and terminate the program.

b. P_4 to P_9 and P_{11} : P_4 to P_7 transform$\gamma\ \beta\ \alpha$to $\gamma\ 00\ \beta\ \alpha$, and on P_8 the instruction in F6 is added to it. As mentioned in section 6d, all instructions in storage are of the form C ⊙ B A, and it will be shown later (sec. 7h) that every instruction is put into F6 prior to its execution. P_9 and P_{11} will store the C⊙B part of the instruction in F6, and leave the isolated A-address in EC (electronic counter).

c. P_{12} to P_{30} : Interpretation of the A-address: By successive subtractions of 1's and balance tests, the machine interprets the A-address, reads out the appropriate general storage unit, and puts A into Fl, 2. Once A has been found, subsequent program steps (in the group P_{12} to P_{30}) are suppressed with the aid of GS # 1 (Group Suppression # 1).

d. P_{31} to P_{52} : Interpretation of the B-address: P_{31} to P_{33} put the B-address into EC and leave C⊙, the remainder of the instruction, in F6. Then (P_{34} to P_{52}) by successive subtractions of 1's and balance tests, the machine interprets the B-address, reads out the appropriate general storage unit, and puts B into F3, 4. Once B has been found, subsequent program steps (in the group P_{34} to P_{52}) are suppressed with the aid of GS# 1.

e. P_{53} to P_{56} : The machine interprets the units position of the operation code. The units position is zero tested (P_{56}), and the result of the test is applied to later steps.

f. P_{57} to P_{114} : The machine interprets the tens position of the operation code, performs the appropriate operation, and, except for logical operations, puts the result (C) into F3, 4.

α. P_{57} to P_{70}: The tens position of the operation code is placed into EC (P_{57} to P_{59}). If the tens position is 0, the operation is subtraction, and the sign of B is changed so that the addition routine can be used later for both addition and subtraction (P_{60} to P_{63}). If it is 1 or 0, the machine adds F1,2 to F3,4 and puts the result (subject to the zero test on P_{56}) into F3,4 (P_{64} to P_{70}).

β. P_{71} to P_{84} : If the tens position of the operation code is 2, the machine multiplies F1,2 by F3,4. The product is rounded (subject to the zero test on P_{56}) and read into F3,4.

γ. P_{85} to P_{98} : If the tens position of the operation code is 3, the machine divides F1,2 by F3,4. The positioning of the dividend is subject to the zero test on P_{56}. The quotient is rounded and read into F3,4.

δ. P_{99} to P_{102} : If the tens position of the operation code is 4, the number in F1,2 is transferred into F3,4 and at the same time shifted one or two places to the left (subject to the zero test on P_{56}).

ϵ. P_{103} to P_{107} : If the tens position of the operation code is 5, the sign of the number in F1,2 is tested. If negative and the units position of the operation code was zero, N.B. selector 1 is picked up. If negative and the units position of the operation code was not zero, N.B. selector 2 is picked up.*

ζ. P_{108} to P_{114} : If the tens position of the operation code is 6, the machine tests whether the number in F1,2 is zero. Subject to the zero test on P_{56}, either the entire number in F1,2 or the entire number except the rightmost position is tested. If zero, N.B. selector 3 is picked up.* Furthermore, F5 is cleared (P_{114}) which will cause any remaining instructions to be skipped.

g. P_{115} to P_{134} : Interpretation of the C-address: By successive subtractions of 1's and balance tests, the machine interprets the C-address and reads F3,4 (C) into the appropriate general storage unit. Once this storage unit has been determined, subsequent program steps (in the group P_{115} to P_{134}) are suppressed with the aid of GS # 1.

* The use of N.B. selectors is explained in the section on the 529 control panel (see sec. 9b).

h. P_{135} to P_{140} : The instructions are "circulated" within the machine by advancing each instruction into the storage unit occupied by its predecessor. The instruction to be executed on the next program sweep is put into F6; this instruction is also moved into F9, 11 (if calculator selector 1 is normal) so that it can be reused (modulo 4). Calculator selector 1 can be picked up on the 529 control panel by an X-punch in col. 35 (see sec. 9b), and if it is transferred, the instruction in F6 is moved into G1, 2 so that it can be reused (modulo 5).

Another comment concerning this group of steps will be found in the Appendix.

8. The 607 Program *

```
 1: F5 RO;         EC  RI+
 2: Emit 1;        EC  RI- ;  B.T.
 3: EC  RR;        F5  RI;    Prog.Rep.;Sup-
 4: F7  RO;        EC  RI+ ;            Sup-
 5: EC  RO;        MQ  RI;    Ro  3rd;  Sup-
 6: MQ  RO;        EC  RI- ;  Ri  3rd;  Sup-
 7: MQ  RO;        EC  RI+ ;  Ri  5th;  Sup-
 8: F6  RO;        EC  RI+ ;            Sup-
 9: EC  RO;        F6  RI;    Ro  2nd;  Sup-
10: Prog.End;                           Sup+
11: F6  RO;        EC  RI- ;  Ri  2nd;
12: Emit 1;        EC  RI- ;  B.T.;     GS#1 d.o.
13: G1,2 RO;       F1,2 RI;   EC  RE;   GS#1 p.u.;   Sup+;
14: Emit 1;        EC  RI- ;  B.T.;     Sup-
15: G3,4 RO;       F1,2 RI;   EC  RE;   GS#1 p.u.;   Sup+;  Sup # 1
16: Emit 1;        EC  RI- ;  B.T.;     Sup-
17: G9,10 RO;      F1,2 RI;   EC  RE;   GS#1 p.u.;   Sup+;  Sup # 1
18: Emit 1;        EC  RI- ;  B.T.;     Sup-
19: G11,12 RO;     F1,2 RI;   EC  RE;   GS# 1 p.u.;  Sup+;  Sup # 1
20: Emit 1;        EC  RI- ;  B.T.;     Sup-
21: G13,14 RO;     F1,2 RI;   EC  RE;   GS#1 p.u.;   Sup+;  Sup # 1
22: Emit 1;        EC  RI- ;  B.T.;     Sup-
23: G15,16 RO;     F1,2 RI;   EC  RE;   GS #1 p.u.;  Sup+;  Sup # 1
24: Emit 1;        EC  RI- ;  B.T.;     Sup-
25: G17,18 RO;     F1,2 RI;   EC  RE;   GS# 1 p.u.;  Sup+;  Sup # 1
26: Emit 1;        EC  RI- ;  B.T.;     Sup-
27: G19,20 RO;     F1,2 RI;   EC  RE;   GS# 1 p.u.;  Sup+;  Sup # 1
28: Emit 1;        EC  RI- ;  B.T.;     Sup-
29: G21,22 RO;     F1,2 RI;   EC  RE;   GS#1 p.u.;   Sup+;  Sup # 1
30: G23,24 RO;     F1,2 RI;   EC  RE;                       Sup # 1
31: F6 RO;         EC  RI+ ;
32: EC  RO;        F6  RI;    Ro  2nd;
33: F6  RO;        EC  RI- ;  Ri  2nd;
34: Emit 1         EC  RI- ;  B.T.;     GS#1 d.o.
35: G1,2 RO;       F3,4 RI;   EC  RE;   GS#1 p.u.;   Sup+
36: Emit 1         EC  RI- ;  B.T.;     Sup-
37: G3,4 RO;       F3,4 RI;   EC  RE;   GS#1 p.u.;   Sup+;  Sup # 1
38: Emit 1         EC  RI- ;  B.T.;     Sup-
39: G9,10 RO;      F3,4 RI;   EC  RE;   GS# 1 p.u.;  Sup+;  Sup # 1
40: Emit 1         EC  RI-;   B.T.;     Sup-
```

* The notation in this and following sections is similar to that employed in Applied Science Newsletter No. 3, pp. 41-42.

41:	G11,12 RO;	F3,4 RI;	EC RE;	GS # 1 p.u.;	Sup+;	Sup # 1
42:	Emit 1;	EC RI- ;	B.T.;	Sup-		
43:	G13,14 RO;	F3,4 RI;	EC RE;	GS # 1 p.u.;	Sup+;	Sup # 1
44:	Emit 1;	EC RI- ;	B.T.;	Sup-		
45:	G15,16 RO;	F3,4 RI;	EC RE;	GS # 1 p.u.;	Sup+;	Sup # 1
46:	Emit 1;	EC RI- ;	B.T.;	Sup-		
47:	G17,18 RO;	F3,4 RI;	EC RE;	GS # 1 p.u.;	Sup+;	Sup # 1
48:	Emit 1;	EC RI- ;	B.T.;	Sup-		
49:	G19,20 RO;	F3,4 RI;	EC RE;	GS # 1 p.u.;	Sup+;	Sup #1
50:	Emit 1;	EC RI- ;	B.T.;	Sup-		
51:	G21,22 RO;	F3,4 RI;	EC RE;	GS # 1 p.u.;	Sup+;	Sup # 1
52:	G23,24 RO;	F3,4 RI;	EC RE;			Sup # 1
53:	F6 RO;	EC RI+ ;				
54:	EC RO;	F6 RI;	Ro 2nd			
55:	F6 RO;	EC RI- ;	Ri 2nd			
56:	Z.T.;	EC RE				
57:	F6 RO;	EC RI+				
58:	EC RO;	F6 RI;	Ro 2nd			
59:	F6 RO;	EC RI- ;	Ri 2nd			
60:	Emit 1;	EC RI- ;	B.T.;			
61:	EC RE;			Sup+		
62:	F3,4 RO;	EC RI- ;		Sup+		
63:	EC RR;	F3,4 RI;		Sup+		
64:	Emit 1;	EC RI- ;	B.T.;			
65:	EC RE;			Sup+		
66:	F1,2 RO;	EC RI+;		Sup+		
67:	F3,4 RO;	EC RI+ ;		Sup+		
68:	EC RR;	F3,4 RI;	Ro 2nd	Sup+;	Sup NZ	
69:	EC RR;	F3,4 RI;		Sup+;	Sup Z	
70:	Skip On;			Sup+		
71:	Emit 1;	EC RI- ;	B.T.			
72:	F3,4 RO;	MQ RI;	EC RE;	Sup+		
73:	F3,4 RO;	EC RI+ ;		Sup+		
74:	EC RR;	F3,4 RI;	Ro 6th;	Sup+		
75:	F1,2 RO;	MPL+;		Sup+		
76:	F3,4 RO;	MQ RI;		Sup+		
77:	EC RR;	F3,4 RI;	Ro 6th	Sup+		
78:	F3,4 RO;	EC RI+		Sup+		
79:	Fl,2 RO;	MPL+;		Sup+		
80:	1/2 Adj;		Ri 3rd	Sup+;	Sup NZ	
81:	EC RR;	F3,4 RI;	Ro 4th	Sup+;	Sup NZ	
82:	1/2 Adj;		Ri 2nd;	Sup+;	Sup Z	
83:	EC RR;	F3,4 RI;	Ro 3rd	Sup+;	Sup Z	
84:	Skip On;			Sup+		
85:	Emit 1;	EC RI- ;	B.T.			
86:	EC RE;			Sup+		
87:	F1,2 RO;	EC RI+ ;	Ri 5th	Sup+;	Sup NZ	
88:	F1,2 RO;	EC RI+ ;	Ri 6th	Sup+;	Sup Z	
89:	F3,4 RO;	DIV.		Sup+		
90:	EC RR;	F1,2 RI;		Sup+		
91:	F1,2 RO;	EC RI+;	Ri 5th	Sup+		
92:	MQ RO;	F1,2 RI;		Sup+		
93:	F3,4 RO;	DIV;	EC RE;	Sup+		
94:	MQ RO;	EC RI+;		Sup+		
95:	F1,2 RO;	EC RI+ ;	Ri 5th	Sup+		
96:	1/2 Adj;			Sup+		
97:	EC RR;	F3,4 RI;	Ro 2nd;	Sup+		
98:	Skip On;			Sup+		

```
 99: Emit 1;     EC RI- ;    B. T.
100: F1,2 RO;    F3,4 RI;    Ri 2nd;    Sup+;          Sup NZ;    EC RE
101: F1,2 RO;    F3,4 RI;    Ri 3rd;    Sup+;          Sup Z;     EC RE
102: Skip On;                           Sup+
103: Emit 1;     EC RI- ;    B. T.
104: EC RE;                             Sup+
105: F1,2 RO;    EC RI+ ;    EC RE;     Sup+;          Sup NZ;    N.B.# 1 p.u.
106: F1,2 RO;    EC RI+ ;    EC RE;     Sup+;          Sup Z;     N.B.# 2 p.u.
107: Skip On;                           Sup+
108: Emit 1;     EC RI- ;    B. T.      EC RE
109: F1,2 RO;    EC RI+ ;               Sup+
110: EC RR;      F1,2 RI;    Ro 2nd;    Sup+;          Sup Z
111: F1,2 RO;    EC RI+ ;               Sup+;          Sup Z
112: Z. T. ;     EC RE;                 Sup+
113: Emit 1;     EC RI- ;               Sup+;          Sup NZ;    N.B.# 3 p.u.
114: F5 RI;                  EC RE      Sup+;          Sup NZ
115: F6 RO;      EC RI+ ;    Skip Off
116: Emit 1;     EC RI- ;    B. T. ;    GS#1 d.o.
117: F3,4 RO;    G1,2 RI;    EC RE;     GS# 1 p.u. ;   Sup+
118: Emit 1;     EC RI- ;    B. T. ;    Sup-
119: F3,4 RO;    G3,4 RI;    EC RE;     GS#1 p.u. ;    Sup+;      Sup# 1
120: Emit 1;     EC RI- ;    B. T. ;    Sup-
121: F3,4 RO;    G9,10 RI;   EC RE;     GS#1 p.u. ;    Sup+;      Sup# 1
122: Emit 1;     EC RI- ;    B. T. ;    Sup-
123: F3,4 RO;    G11,12 RI;  EC RE;     GS# 1 p.u. ;   Sup+;      Sup# 1
124: Emit 1;     EC RI- ;    B. T. ;    Sup-
125: F3,4 RO;    G13,14 RI;  EC RE;     GS# 1 p.u. ;   Sup+;      Sup# 1
126: Emit 1;     EC RI- ;    B. T. ;    Sup-
127: F3,4 RO;    G15,16 RI;  EC RE;     GS#1 p.u. ;    Sup+;      Sup# 1
128: Emit 1;     EC RI-      B. T. ;    Sup-
129: F3,4 RO;    G17,18 RI;  EC RE;     GS #1 p.u. ;   Sup+;      Sup# 1
130: Emit 1;     EC RI- ;    B. T. ;    Sup-
131: F3,4 RO;    G19,20 RI;  EC RE;     GS #1 p.u. ;   Sup+;      Sup# 1
132: Emit 1;     EC RI- ;    B. T. ;    Sup-
133: F3,4 RO;    G21,22 RI;  EC RE;     GS #1 p.u. ;   Sup+;      Sup# 1
134: F3,4 RO;    G23,24 RI;  EC RE;                               Sup# 1
135: F8 RO;      F6 RI
136: F10 RO;     F8 RI
137: F12 RO;     F10 RI
138: F9,11 RO;   F12 RI
139: G1,2 RO;    F9,11 RI;    Sup w/o Test (Cal-Sel 1 N)
140: F6 RO;      F9,11 RI (Cal-Sel 1 N); G1,2 RI (Cal-Sel 1 T)
```

Storage Assignments

"8-6": G1,3,9,11,13,15,17,19,21,23, F1,3.

"6-4": F9

NOTE: Program expanders are used whenever more than four program exits are required.

9. The 529 Control Panel

This section outlines the main features of the 529 control panel. A review of sections 5 and 6 will show the card fields that are wired to various 607 storage entries and exits. A summary is listed below.

a. Entries and Exits.

n → F5* ; C.C. → F5 RI
$\alpha\beta\gamma$ → F7* ; C.C. → F7 RI

Instruction 1 → F6*
" 2 → F8*
" 3 → F10*
" 4 → F12*
" 5 → F9,11*

Instructions are wired through the normal side of a group of selectors picked up by Y_1 **. The F6,8,9,10,11,12 RI hubs are impulsed by C.C. also through the normal side of these selectors. Through the transferred side of the selectors, three 8-position input-data card fields are wired to the entries of G13,14; G15,16; and G17,18, respectively. X-punches in the leftmost position of each field are wired to the RI hubs of the corresponding storage units (for further details see sec. 5b):

Input-data → G1,2* ; X_{33} → G1,2 RI
" → G3,4 ; X_{41} → G3,4 RI
" → G9,10 ; X_{49} → G9,10 RI
" → G11,12; X_{57} → G11,12 RI
G21,22 → punching hubs; C.C. → G21,22 RO*
G23,24 → punching hubs; C.C. → G23,24 RO*
To clear the counter, wire C.C. → EC RR

b. Selectors

G21,22 RO is wired through the transferred side of a pilot selector picked up by X_{65}.

G23,24 RO is wired through the transferred side of a pilot selector picked up by X_{73}.

Y_1 (special data card) is wired to the I-p.u. of a group of punch selectors.

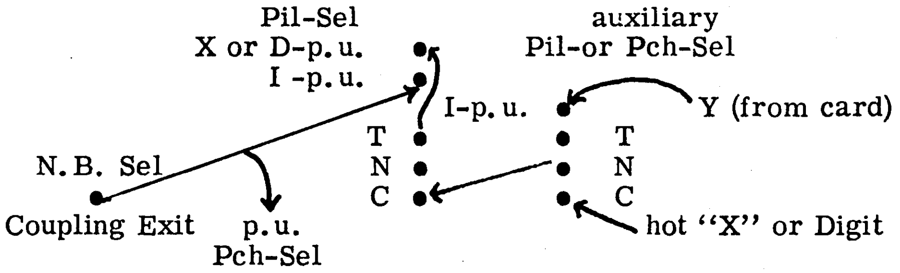
The use of these selectors has already been explained (see sec. 9a).

X_{35} is wired to the p.u. hub of calculator selector 1, which permits the use of G1,2 for a sixth instruction. It will be recalled (see sec. 6d) that when an instruction is read into storage its A-address and C-address are interchanged; a punch selector, picked up by X_{35}, is used to transpose the leftmost and rightmost position of the sixth instruction.

* Through selectors as explained in section 9b.

** Y_i means Y-punch in column i.

Latch selectors: As explained in section 7, N.B. selectors 1,2 and 3 may be picked up as a result of certain negative or zero conditions. Each N.B. selector is used to transfer a pilot selector. With the aid of an auxiliary selector, the pilot selector is held in a transferred position until a card with a special Y-punch is read. The pilot selector will also be transferred for the Y-card, but, thereafter, it will return to its normal position. Punch selectors can be coupled to the pilot selector. The diagram below illustrates a method for wiring a typical "holding circuit."



We shall refer to the group of selectors associated with N.B. selectors 1,2,3 as latch selectors I, II, III, respectively. Y_5, Y_6, Y_7 pick up the auxiliary selectors and initiate the "drop out" of latch selectors I, II, III, respectively.

Latch selector I is used to prevent the entry of instructions 1 and 2 into F6 and F8. In their place instructions 3 and 4 are entered into F6 and F8. (See Appendix for further remarks regarding the selection of instructions.)

Latch selector II permits the entry of $\alpha\beta\gamma$ into F7, i.e., $\alpha\beta\gamma$ is wired through the transferred side of latch selector II. Since the RI hub of F7 is impulsed by C.C. all the time, F7 will normally be zero.

Latch selector III prevents the entry of n into F5, i.e., n is wired through the normal side of latch selector III. This is equivalent to reading n=0 into F5 whenever latch selector III is transferred, since the RI hub of F5 is always impulsed by C.C.

10. Appendix

a. Suggestions for modifications.

Unlike data entry, which is under the control of individual X-punches, instructions are read automatically into storage. The card-cycles pulse, which is responsible for the automatic entry of all instructions, could be replaced by X-punches to control the entry of each instruction individually. The user would have to decide whether the flexibility gained thereby would compensate for the inconvenience of punching additional X's into the card.

In the present set-up, only two results may be punched out from G21, 22 and G23, 24. This number could be increased by the proper selection of data input and output.

The selection of instructions by means of latch selector I need not be done entirely on the 529 control panel. Several punch selectors may be eliminated by the association of a single calculator selector, say Cal-Sel 2, with latch selector I. Only the selection between instruction 1 (through the normal side of latch selector I) and instruction 3 (through the transferred side of latch selector I) has to be done on the 529. The selection of the remaining instructions may be accomplished by the following minor modification in the 607 program.

P_{137} : F12 RO; F10 RI (Cal-Sel 2 N); F6 RI (Cal-Sel 2 T)

For maximum speed, Sup w/o Test (Cal-Sel 2 T) should be added to P_{135} and P_{136}.

b. Suggestions for additional operations and instructions.

With the aid of calculator selectors, additional operations could be incorporated. Selected program exits can be used for multiple purposes, and an effort should be made to design the 607 control panel so that a group of steps can be used for more than one operation even without selection. However, since calculator selectors stay picked up for the entire calculation, the coder would be required to exclude on any given card any instructions that involve both the normal and transferred positions of a calculator selector. If selectors which could be picked up and dropped out on different program sweeps, such as the program repeat selectors on the 605, were available, program exits could be selected so as to produce more than 140 distinct program steps.

The number of instructions that may be read from a single card depends primarily on the storage capacity of the machine. Each instruction requires at least one 5-position storage unit. In the preceding sections it has been shown how a storage unit can be used alternatively for data and instructions. Consequently, more instructions may be included at the expense of data storage. The ease with which additional instructions may be incorporated in the 607 program is evident from a study of program steps 135-140. Only a single additional program step is required for each additional instruction.

AN ITERATIVE TECHNIQUE FOR MULTIPLE CORRELATION ANALYSIS

Martin H. Greenberger *
Joe H. Ward, Jr.

Air Force Personnel and Training Research Center
Lackland Air Force Base
San Antonio, Texas

I. Introduction

An earlier iterative method by the same authors appeared in IBM Technical Newsletter No. 6 and may be referred to as background for this paper. The same notation has been maintained as much as possible. One exception is that V_i has been replaced by r_{i0} to represent the ith criterion correlation coefficient.

Although the approach here is somewhat different, the problem is identical with the one treated by the earlier method. It is desired to "fit" (in the least squares sense) a hyper-plane in (n+1) space to the set of N points

$$\begin{array}{c}(Z_{10}, Z_{11}, Z_{12}, \ldots, Z_{1n}) \\ (Z_{20}, Z_{21}, Z_{22}, \ldots, Z_{2n}) \\ \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \\ (Z_{N0}, Z_{N1}, Z_{N2}, \ldots, Z_{Nn})\end{array}$$

in order to permit prediction of the 0th dimension coordinate from the coordinates of the other n dimensions. That is, given any arbitrary point $(Z_{m1}, Z_{m2}, \ldots, Z_{mn})$, a least squares set of weights $B_1, B_2, \ldots, B_n$ is desired which will predict Z_{m0} by the equation

$$Z_{m0} = B_1 Z_{m1} + B_2 Z_{m2} + \ldots + B_n Z_{mn}$$

In practice, N will be much larger than n. Hence by replacing the original N points with $B_1, B_2, \ldots, B_n$, considerable data reduction is obtained.

Both iterative methods begin with a set of weights $X_1, X_2, \ldots, X_n$ which fail to yield the desired fit. By successive corrections on one X at a time the fit is approached. If

* Now with the Harvard Computation Laboratory.
Note: 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.

$$L_1 = X_1 Z_{11} + X_2 Z_{12} + \dots + X_n Z_{1n}$$

$$L_2 = X_1 Z_{21} + X_2 Z_{22} + \dots + X_n Z_{2n}$$

$$\cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot$$

$$L_N = X_1 Z_{N1} + X_2 Z_{N2} + \dots + X_n Z_{Nn} ,$$

the desired fit can be achieved by reducing the differences between the L_i and the Z_{0i} with the appropriate succession of corrections.

The earlier method converged on the solution by choosing at each stage a correction on some selected X which minimized

$$\sum_{i=1}^{N} (Z_{i0} - L_i)^2$$

with all other X held constant. In contrast, the present method chooses that correction for the selected X which maximizes the correlation between $(L_1, L_2, \dots, L_N)$ and $(Z_{10}, Z_{20}, \dots, Z_{N0})$, all other X fixed. In both cases, the derived least squares weights are labeled $B_1, B_2, \dots, B_n$.

The former iterative method afforded the utmost in simplicity computationally, and was therefore ideal for 602-A application. The present method involves more sophisticated computation but offers faster convergence. In theory any number of variables may be treated by this technique. The present program handles up to 75 variables.

The iterative technique to be developed was designed for the 607 Electronic Calculator. However, with some revision it may be adapted to the 650. The present approach eliminates the necessity for panel wiring changes during the operation. From iteration to iteration there is completely automatic field selection. It has been in successful operation since June 1954, and has provided precious time savings during this period. Section VII discusses one of the representative problem solutions.

The procedure is an offspring of the Kelley-Salisbury method* with some important modifications. For example, it is not satisfactory to correct at each stage on the equation with the largest numerical error, as the Kelley-Salisbury method suggests. It is wholly within reason for a variable to produce the greatest error without admitting of an appreciable correction by the criterion used. One further modification is the punching of e_i (defined below) instead of a_i. The former has the advantage of being restricted to a narrow range of magnitude.

* Kelley, T. L. and Salisbury, F. S., "An Iteration Method for Determining Multiple Correlation Constants," Journal of the American Statistical Association, 21:282-292, 1926.

It would be well at this point to introduce some additional notation which will be used throughout the ensuing development. The following relations are satisfied by the least squares weights:

$$\begin{array}{l} B_1 + r_{12} B_2 + \dots + r_{1n} B_n = r_{10} \\ r_{12} B_1 + \quad B_2 + \dots + r_{2n} B_n = r_{20} \\ \dots\dots\dots\dots\dots\dots\dots \\ r_{1n} B_1 + r_{2n} B_2 + \dots + \quad B_n = r_{n0} \end{array}$$

where r_{ij} = the correlation between Z_i and Z_j

$r_{ij} = r_{ji}$ and $r_{ii} = 1$.

For the intermediate weights $X_1, X_2, \dots, X_n$ the following a_i are defined:

$$\begin{array}{l} X_1 + r_{12} X_2 + \dots + r_{1n} X_n = a_1 \\ r_{12} X_1 + \quad X_2 + \dots + r_{2n} X_n = a_2 \\ \dots\dots\dots\dots\dots\dots\dots \\ r_{1n} X_1 + r_{2n} X_2 + \dots + \quad X_n = a_n \end{array}$$

Unless $X_1, X_2, \dots, X_n$ is identical with $B_1, B_2, \dots, B_n$ the a_i will not in general equal the r_{i0}. Letting $a_i - r_{i0} = e_i$, and calling this difference the ith error, the condition for solution is that the e_i approximate zero for all i = 1 to n.

II. Mathematical Statement

A. We are given the n + 1 variables $z_0, z_1, z_2, \dots, z_n$, where $Z_i = (Z_{i1}, Z_{i2}, \dots, Z_{in})$ is a set of N standardized measurements such that

$$\Sigma z_{0k} = \Sigma z_{1k} = \Sigma z_{2k} = \dots = \Sigma z_{nk} = 0$$

and

$$\frac{1}{N} \Sigma z_{0k}^2 = \frac{1}{N} \Sigma z_{1k}^2 = \frac{1}{N} \Sigma z_{2k}^2 = \dots = \frac{1}{N} \Sigma z_{nk}^2 = 1.$$

We compute the correlation coefficients *

$$r_{ij} = r_{ji} = \frac{1}{N} \Sigma z_{ik} z_{jk} \qquad \text{(i and j = 0, 1,\dots, n)}$$

As a first approximation toward predicting z_0 from $z_1, z_2, \dots, z_n$, we form the linear combination

* Throughout the development the summation Σ will always be over k = 1 to N. From here on, however, the k subscript will be omitted for notational simplicity.

$$L = x_1 z_1 + x_2 z_2 + \dots + x_n z_n ,$$

where $x_1, x_2, \dots, x_n$ is any set of scalar constants. Later, the x become the variables of simultaneous linear equations and should not be confused with the z which were first introduced as variables. The z are more precisely vectors; each z might, for example, be a set of test scores. In the final correlation analysis the z become implicit and the mathematics uses only their interrelations, the r_{ij}.

B. The correlation between z_0 and L is derived as follows:

$$r_{0L} = \frac{\frac{1}{N} \Sigma (z_0 - m_0)(L - m_L)}{\sigma_0 \sigma_L}$$

where

$$m_0 = \frac{1}{N} \Sigma z_0 = 0$$

$$\sigma_0 = \sqrt{\frac{1}{N} \Sigma (z_0 - m_0)^2} = 1$$

$$m_L = \frac{1}{N} \Sigma L = \frac{1}{N} \Sigma (x_1 z_1 + x_2 z_2 + \dots + x_n z_n)$$

$$= x_1 \frac{1}{N} \Sigma z_1 + x_2 \frac{1}{N} \Sigma z_2 + \dots + x_n \frac{1}{N} \Sigma z_n$$

$$= x_1 (0) + x_2 (0) + \dots + x_n (0)$$

$$= 0$$

so that

$$r_{0L} = \frac{\frac{1}{N} \Sigma z_0 L}{\sigma_L}$$

Tagging r_{0L} with the new name MCC, dropping the subscript of σ_L, and letting

$$S = \frac{1}{N} \Sigma z_0 L = \frac{1}{N} \Sigma z_0 (x_1 z_1 + x_2 z_2 + \dots + x_n z_n)$$

$$= x_1 \frac{1}{N} \Sigma z_0 z_1 + x_2 \frac{1}{N} \Sigma z_0 z_2 + \dots + x_n \frac{1}{N} \Sigma z_0 z_n$$

$$= x_1 r_{10} + x_2 r_{20} + \dots + x_n r_{n0}$$

provides the final relation

$$MCC = \frac{S}{\sigma}$$

C. It is now necessary to derive an expression for σ or more conveniently, for σ^2:

$$\sigma^2 = \frac{1}{N} \Sigma L^2 \qquad \text{(since } m_L = 0\text{)}$$

$$= \frac{1}{N} \Sigma (x_1 z_1 + x_2 z_2 + \dots + x_n z_n)^2$$

$$= x_1 \frac{1}{N} \Sigma z_1 (x_1 z_1 + x_2 z_2 + \dots + x_n z_n) + \dots + x_n \frac{1}{N} \Sigma z_n (x_1 z_1 + \dots + x_n z_n)$$

$$= x_1 (x_1 \frac{1}{N} \Sigma z_1{}^2 + x_2 \frac{1}{N} \Sigma z_1 z_2 + \dots + x_n \frac{1}{N} \Sigma z_1 z_n) + \dots + x_n (x_1 \frac{1}{N} \Sigma z_1 z_n + \dots + x_n \frac{1}{N} \Sigma z_n{}^2)$$

$$= x_1 (x_1 + x_2 r_{12} + \dots + x_n r_{1n}) + \dots + x_n (x_1 r_{1n} + \dots + x_n)$$

$$= x_1 a_1 + x_2 a_2 + \dots + x_n a_n$$

The expanded representation of MCC or, more conveniently, of $(MCC)^2$ is thus

$$(MCC)^2 = \frac{(x_1 r_{10} + x_2 r_{20} + \dots + x_n r_{n0})^2}{x_1 a_1 + x_2 a_2 + \dots + x_n a_n}$$

D. An important question is which set or sets of $x_1, x_2, \dots, x_n$ gives a maximum $(MCC)^2$. This is answered by partial differentiation as follows:

$$\frac{\partial}{\partial x_j} (MCC)^2 = \frac{\partial}{\partial x_j} \frac{S^2}{\sigma^2} = \frac{\sigma^2 \frac{\partial}{\partial x_j} S^2 - S^2 \frac{\partial}{\partial x_j} \sigma^2}{\sigma^4}$$

But $$\frac{\partial}{\partial x_j} S^2 = \frac{\partial}{\partial x_j} (x_1 r_{10} + \dots + x_n r_{n0})^2$$

$$= 2 S r_{j0}$$

and $$\frac{\partial}{\partial x_j} \sigma^2 = \frac{\partial}{\partial x_j} (x_1 a_1 + \dots + x_n a_n)$$

$$= (x_1 \frac{\partial}{\partial x_j} a_1 + \dots + x_j \frac{\partial}{\partial x_j} a_j + \dots + x_n \frac{\partial}{\partial x_j} a_n) + a_j$$

Since $$a_i = x_1 r_{i1} + \dots + x_j r_{ij} + \dots + x_n r_{in},$$

$$\frac{\partial}{\partial x_j} a_i = r_{ij}$$

and $$\frac{\partial}{\partial x_j} \sigma^2 = (x_1 r_{j1} + x_2 r_{j2} + \dots + x_n r_{jn}) + a_j$$

$$= 2 a_j$$

finally, $\frac{\partial}{\partial x_j}(MCC)^2 = \frac{2S\sigma^2 r_{j0} - 2S^2 a_j}{\sigma^4}$

If this is set equal to zero, it is seen that the minimum $(MCC)^2$ occurs when S, and hence $(MCC)^2$, equals zero, and the maximum when

$$\frac{S}{\sigma^2} a_j = r_{j0} \qquad (j = 1, 2, \ldots, n)$$

But the least squares solution $B_1, B_2, \ldots, B_n$ is such that

$$a_j = r_{j0}$$

This states that the solution obtained by maximizing $(MCC)^2$ is just a constant multiple σ^2/S times the least squares solution. Now S and σ^2 are immediately available for any $x_1, x_2, \ldots, x_n$. Therefore, once an $x_1, x_2, \ldots, x_n$ is derived which maximizes $(MCC)^2$, the least squares solution can be quickly computed by the equation

$$B_i = X_i \left(\frac{S}{\sigma^2}\right)_{\text{final}}$$

E. It remains only to outline the technique for converging upon the maximum $(MCC)^2$. One variable at a time is corrected upon. The correction is such that a maximal increase in $(MCC)^2$ is effected. This correction is derived as follows, assuming x_P is the variable to be corrected upon:

If δ_P is the correction such that the new $x_P = x_P' = x_P + \delta_P$,

then: $$\begin{aligned} \text{the new } S = S' &= x_1 r_{10} + \ldots + (x_P + \delta_P)\, r_{P0} + \ldots + x_n r_{n0} \\ &= (x_1 r_{10} + \ldots + x_P r_{P0} + \ldots + x_n r_{n0}) + \delta_P r_{P0} \\ &= S + \delta_P r_{P0}, \end{aligned}$$

$$\begin{aligned} \text{the new } a_i = a_i' &= x_1 r_{i1} + \ldots + (x_P + \delta_P)\, r_{iP} + \ldots + x_n r_{in} \\ &= (x_1 r_{i1} + \ldots + x_P r_{iP} + \ldots + x_n r_{in}) + \delta_P r_{iP} \\ &= a_i + \delta_P r_{iP}\,, \end{aligned}$$

$$\begin{aligned} \text{and the new } \sigma^2 = (\sigma^2)' &= x_1 a_1' + \ldots + (x_P + \delta_P)\, a_P' + \ldots + x_n a_n' \\ &= (x_1 a_1 + \ldots + x_n a_n) + \delta_P\,(x_1 r_{P1} + \ldots + x_n r_{Pn}) + \delta_P (a_P + \delta_P) \\ &= \sigma^2 + \delta_P a_P + \delta_P a_P + \delta^2{}_P \\ &= \sigma^2 + 2_{aP}\delta_P + \delta^2{}_P \end{aligned}$$

Now, if $(MCC)^{2'}$ is the new $(MCC)^2$ under a change δ_P in x_P ,

$$\frac{\partial}{\partial\delta_P}(MCC)^{2'} = \frac{\partial}{\partial\delta_P} \frac{(S')^2}{(\sigma^2)'}$$

$$= \frac{(\sigma^2)' (2S' \frac{\partial}{\partial\delta_P} S') - (S')^2 \frac{\partial}{\partial\delta_P}(\sigma^2)'}{[(\sigma^2)']^2}$$

It should be emphasized that the partial differentiation is with respect to δ_P , the correction in x_P to be determined.
Using the fact that

$$\frac{\partial}{\partial\delta_P} S' = \frac{\partial}{\partial\delta_P} (S + \delta_P r_{P0}) = r_{P0}$$

and
$$\frac{\partial}{\partial\delta_P}(\sigma^2)' = \frac{\partial}{\partial\delta_P} (\sigma^2 + 2a_P\delta_P + \delta^2_P) = 2 (a_P + \delta_P)$$

we see that

$$\frac{\partial}{\partial\delta_P}(MCC)^{2'} = \frac{2r_{P0}S' (\sigma^2)' - 2 (a_P + \delta_P) (S')^2}{[(\sigma^2)']^2}$$

Substituting in the derived expressions for S' and $(\sigma^2)'$ and setting this equal to zero gives the condition that

$$\delta_P = \frac{\sigma^2 r_{P0} - Sa_P}{S - a_P r_{P0}}$$

Now δ_P is just the correction in x_P that maximizes $(MCC)^2$ with all other x_i held constant.

As this process is repeated, each time correcting upon another x_i, $(MCC)^2$ constantly increases and must approach the desired maximum. The method is therefore convergent.

The criterion for determining P at each stage is as follows: A δ is calculated for each of the n x, using the S, σ^2, and the set of a that were computed after the prior correction. The largest numerical δ is then selected as δ_P and the corresponding x_P corrected upon by this amount. A new S and σ^2 is calculated, and on the next iteration all e_i are adjusted.

$$e_i' = e_i + \delta_P r_{iP}$$

Since $$a_i' = r_{i0} + e_i' ,$$

the new set of δ can be computed simultaneously with the adjustment of the e_i.

Besides an S and a σ^2, an $(MCC)^2$ is computed at the completion of each iteration. Whenever the difference between successive $(MCC)^2$ falls below an established tolerance, iterations are suspended. The $x_1, x_2, \ldots, x_n$ are obtained by summing the δ_P, and they are multiplied by the ratio of the final S to the final σ^2 to give $B_1, B_2, \ldots, B_n$, the desired regression or least squares solution.

It is interesting to note that when $B_1, B_2, \ldots, B_n$ replaces the general $x_1, x_2, \ldots, x_n$, the following relations exist:

$$S = \sigma^2 = (MCC)^2$$

$$a_i = r_{i0} \qquad (i = 1, 2, \ldots, n)$$

and $$e_i = 0$$

III. Table of Notation

Symbol	Meaning	Explanation	Possible Values
prc	proper r card	which of three r cards to be read from	0, 1, 2
rc	r card	number of r card going through	0, 1, 2
e#	e code number	setting up selectors on punch	
r#	r code number	setting up selectors on punch	
mr	matrix row	for e cards and r cards	00 thru n-1 for matrix of order N
ef	e field	field of card from which e is read	00 thru 24
mc	matrix column	column to be read from on a given run	00 thru n-1
X_7e	e card, with X_7	exhausted e card; new e punched on X_7'	
$X_7'e$	e card, following X_7	blank e card	
X_6e	e card with X_6	dummy e card	

X_i designates that an 11-punch appears in card column i.

Y_i designates that a 12-punch appears in card column i.

IV. Setup of the Decks

F δ Card

The Fδ card is the first card through the machine on each iteration and contains all of the data that is constant for calculations on the remaining cards. The Fδ card contains the following data:

Card Column	Data
1 - 2	Variable identification 00 through n-1
3 - 4	e code number
4	12-punch to designate an Fδ card
5 - 9	δ_P
10 - 14	S
15 - 19	σ^2
20 - 24	$(MCC)^2$
79 - 80	iteration number

The Fδ card must be key-punched to start the problem. All succeeding Fδ cards are punched by the 607. The variable having the largest r_{i0} is corrected first, and consequently the initial Fδ card appears as follows:

Card Column	Quantity	Which is denoted
1 - 2	i	mc
3 - 4	00	ef
4	12	identifies Fδ
5 - 9	r_{i0}	δ
10 - 14	r_{i0}^2	S
15 - 19	r_{i0}^2	σ^2
20 - 24	r_{i0}^2	$(MCC)^2$
79 - 80	00	Iteration number

It is important to remember that the initial i corresponds to the row which has the largest numerical r_{i0}.

r Cards

The r cards contain the elements r_{ij}, with each card holding 25 three-digit elements of a row of the matrix. The present machine procedure provides for three r-cards or 75 variables. The r card contains the following data:

Card Column	Data
1 - 2	Matrix row number, 00 through n-1
2	12-punch to indicate an r card
3 - 77	25 r of a given row, 3 digits each
78 - 80	r_{i0}

If n >25, more than one r card must be used for each row of the matrix. This requires the punching of more than one r deck. Twenty-five coefficients should be punched on the first deck, twenty-five on the second, and so on, until all have been punched. Since three positions are allowed for each r value, it is necessary to use .999 as the diagonal elements rather than 1.000. This has no appreciable effect on the final results.

When more than one r card is required for each row, care should be exercised in adhering to the following codes for the various r cards:

	Second from Last r card	Next to Last r card	Last r card
One r-card problem			no additional punch
Two r-card problem		X_6	no additional punch
Three r-card problem	X_7	X_6	no additional punch

e Cards

The e cards are used for punching the successive errors on each iteration. A value e_i is read from the ith e card and e_i' is punched in the adjacent card field of the same card. All e cards have the following card form:

Card Column	Data
1 - 2	matrix row number, 00 through n-1
3 - 5	$-r_{i0}$ (first e deck only)
3	12-punch to identify as an e card

There must be a "dummy" e card behind the Fδ card. This "dummy" e contains an 11-punch in column 6 in addition to the 12-punch in column 3. This card is required to set up selectors for reading the first r card.

The first set of e cards will contain the errors which result from first guesses of zeros for all weights. This results in errors $e_i = -r_{i0}$, and these are punched in columns 3 - 5 as noted above. From then on the 607 reads e_i from columns 3 - 5 and punches e_i' in columns 6 - 8, reads from 6 - 8 and punches in 9 - 11, and so on, until 25 errors are punched on a card. On the 25th iteration (iterations are numbered 00-24) the machine punches an X_7 in the e cards and the machine stops. This indicates to the operator that the e cards are exhausted and that a new e deck of X_7' must be inserted. This X_7' deck is sorted or merged behind the X_7 e deck. On the next run the e_i are read from the last field of the exhausted e deck and the new e_i are punched into columns 3 - 5 of the new deck. The old e cards are then removed and the iterations are continued. When a third e deck is required, the exchange process is repeated.

Lδ Cards

The Lδ cards are blank cards containing the iteration number in columns 79-80. One Lδ card is punched at the end of each iteration, and an Lδ card for the kth iteration becomes the Fδ card for the (k + 1)st iteration.

B Cards

These cards are blank and contain the variable number in columns 1 - 2. After the iterations have been completed, the final B_1, B_2, ..., B_n are punched in the B cards and are ready for listing.

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 Fδ card (number 00).
2. A dummy e card containing an X_6 in addition to the Y_3.
3. The r cards from the first row (row 00) of the matrix. Care must be taken to have the proper code punches as indicated previously. For a three r-card problem the first r card would contain an E_7, the second r card would contain an X_6, and the last r card would contain no code punches except the Y_2 which identifies all r cards.
4. The e card for the first row.
5. Three and four above are repeated for all other rows.
6. The Lδ card. It will become the new Fδ card.

After every iteration a card moves from the stack of blank Lδ cards to the bottom of the operational deck; the former Lδ card becomes Fδ and the former Fδ card goes to the stack of used Fδ cards. No control panel wiring changes are necessary throughout the entire operation. The machine automatically selects the appropriate field from each card, as directed by the r# and e#.

When the e cards become exhausted, they will be replaced as previously described; the iterative procedure then continues normally.

The method will cause $(MCC)^2$ to increase at each iteration, and it can be stopped whenever the desired tolerance is achieved. As soon as $|(MCC)^2 - (MCC)^{2'}|$ is within this tolerance, the machine will pick up calculator selector 13 and punch an X into column 1 of the Lδ card. The operator may check the Lδ and Fδ cards to determine if $|(MCC)^2 - (MCC)^{2'}|$ is within the predetermined tolerance.

The machine now places S/σ^2 into the MQ unit. This constant is used to obtain $B_1, B_2, \ldots, B_n$ from $x_1, x_2, \ldots, x_n$. It is necessary to inspect the MQ unit to see that it does contain the constant S/σ^2. If the MQ unit is blank, the quotient $S/\sigma^2 > 1$, and the calculate panel must be changed to allow for one whole number and four decimals. This is done by making the following changes:

Program	
102	Change Read Units Into 5 to Read Units Into 4.
103	Change Read Units Out of 6 to Read Units Out of 5.
137	Change Read Units Into 6 to Read Units Into 5.

After the three wire changes are made, the punch panel is opened to drop out calculator selector 13 and the last iteration is repeated to recompute the constant S/σ^2. As soon as calculator selector 13 is picked up and S/σ^2 is located in the MQ unit, the least squares weights may be computed.

In order to obtain the final desired weights, it is necessary to sort the δ cards in variable number order with a blank B card behind each of the corresponding group of δ cards. This deck is then run through the 607, which has been set up by calculator selector 13, to compute the sum of δ corrections for each variable and multiply by S/σ^2 to obtain the B weights. The B weights are then removed from the deck and are ready for listing. It may be desired to check computations by substituting the B weights into the equations. This can be done by using the B weights as a predetermined set of $F\delta$ cards and placing a blank r card (Y_2 card) at the end to keep calculator selector 13 from transferring. Thus the checking will require as many iterations as there are non-zero B weights. At the end of the last iteration, the last field that is punched in each e card should be approximately equal to zero.

VI. Description of Programming

Section VI comprises a detailed program of the procedure for the Type 607, together with auxiliary charts and tables. It is too bulky for inclusion in this article, but will be made available upon request.*

VII. Solution of a 24th Order Regression Weight Problem

As was pointed out earlier, up to 75 variables can be treated by the program without further modification. With some changes this upper limit may be raised arbitrarily high. In general, convergence will be slowed down somewhat as the size of the

* Requests for the programming description should be directed to:

The Applied Science Division,
International Business Machines Corporation,
590 Madison Avenue, New York 22, N.Y.

problem increases.

To provide an indication as to the convergence time of this method, a representative problem of the 24th order has been selected for discussion. Thirty-two iterations were required for convergence to an MCC, constant to four places. When the derived set of B values were substituted back in the given equations, the magnitude of the largest error was .007. The overall running time was under an hour. In general, the amount of time required is a direct function of the number of iterations necessary for the desired accuracy. Since the MCC is computed as part of the machine procedure after each iteration, an immediate measure of convergence is always automatically available.

ACKNOWLEDGMENT

Considerable credit for the completion of this report is due Mr. Raymond Martinez, who supervised the computation at the operational stage.

AUTOMATIC DIAGNOSIS OF CPC CODING ERRORS

D. W. Marquardt
E. I. du Pont de Nemours and Company, Inc.
Wilmington, Delaware

Introduction

One of the major costs in the use of digital computers for technical computation is that of ''debugging'' new programs. Accordingly, the development of systematic procedures for the diagnosis of coding errors offers the possibility of significantly reducing computation costs. A description will be given here of some techniques that have been in routine use with marked success for more than a year. These techniques, although designed for use with an adaptation of the Heising floating decimal setup, are applicable with only minor modifications to many of the general purpose setups currently used with the Model II CPC. These setups typically provide for up to three operands, leading to a single result, on each card cycle.

Classification of Errors

In general, there are two broad classes of errors that may arise in the preparation of problems for a digital computer. First, there are logical errors in planning the strategy of problem solution. Errors of logic are not discussed here.

Secondly, there are coding errors which arise in the transcription of the detailed strategy of solution into computer code. CPC coding errors may be grouped conveniently into two types as follows:

1. Type 1 Errors

   These errors result from violations of built-in limitations of the CPC (with a typical setup), namely:

   (a) The addresses specified for the three operands on any card must reflect the limitations that not more than two operands can come from 941-type storage, nor more than two from accounting machine storage.

   (b) The addresses specified for the three operands and for the result must satisfy the timing restrictions, which are as follows:

   (1) It is not possible to read an operand out of an accounting machine storage location and read the result back into that same location. It is possible, however, to do this with 941-type storage locations.

(2) The result of the (n - 1)st card, if sent to an accounting machine storage location, may not be obtained from that location as an operand on the nth card.

(3) The result of either the (n - 2)nd or (n - 1)st card, if sent to a 941-type storage location, may not be obtained from that location as an operand on the nth card.

(4) The results from two successive cards may not be sent to the same storage location.

2. Type 2 Errors

This category includes all coding errors not due to the limitations listed above.

Significance of Type 1 Errors

A major significance of Type 1 errors is that they are readily susceptible to automatic diagnosis. Their automatic diagnosis is made more important by the fact that some Type 1 errors will not always, or even most of the time, cause the CPC to malfunction. Consequently, they will frequently escape detection during normal debugging of a code but manifest themselves during the computational phase. The remainder of this article will describe techniques for the automatic diagnosis of Type 1 errors.

Techniques of Diagnosis

It is possible to check for the presence of coding errors by means of a special run or runs (on the CPC or on some other machine*) prior to the computation runs. It is alternatively possible to have the CPC perform at least some of these checks continuously as the computations proceed.

There is much to recommend the latter procedure. The CPC is frequently employed for engineering-type calculations. These are often characterized by the self-service use of the CPC by non-specialists, and by changes in problem complexion as the results develop, necessitating changes in strategy and hence changes in coding. Experience has shown that coding errors are somewhat more frequent in coding modifications than in original coding. Therefore, a continuous check is desirable, provided it is not necessary to sacrifice a significant part of computing capacity in order to obtain it.

The present procedure is a compromise between prechecking and continuous checking. A continuous check is made for Type 1-a errors at every card cycle.

*Note: The present procedures employ the CPC exclusively. A control panel to perform some of the checks described here has been developed using the Type 101 Statistical Machine (1). Some of the checks described here have also been included in the Rand Corporation "Duplex" setup (2).

A special 605 panel and a 527 panel to go with it are used to make a precheck for all timing errors (Type 1-b) in a single pass of the instruction cards through the 527 punch unit.

It will be assumed in what follows that not more than 6 banks of 941 storage are present and that accounting machine storage has addresses 91, 92, ..., 98. The manner in which the system should be modified for variations of this will be obvious.

Control Panel Wiring to Check for Type 1-a Errors

The accounting machine panel wiring to perform this check is shown in Figure 1. The CPC will stop and the stop light will turn on if any card fails to meet the requirement that there be at least one "9" but not more than two "9's" in the tens positions of the addresses for the three operands. Obviously, it will be necessary to punch a dummy "9" in any blank cards. To restart the CPC, the final total button and then the start button must be pressed.

It will be noted that relatively few components are used to perform the check, so that there is no significant loss in computing capacity.

Control Panels to Check for Type 1-b Errors

These panels will automatically check a deck of instruction cards for the presence of all Type 1-b errors (timing restriction violations). It is recommended that the cards be passed through this timing check procedure before debugging is begun, and again after it is otherwise completed. The panels (527 and 605) require only the 605 unit and the 527 unit of the CPC. The cards are passed through the 527 unit, and if no timing errors are present, the cards will not stop. However, if a timing error is present, the zero check light on the 527 unit will light, and subsequently the cards will cease to feed. The last card to be ejected into the stacker before the cards stop feeding contains the timing error.* If an error has been detected, so that the zero check light is lit, then the reset button must be pressed before the start button can cause feeding to resume.

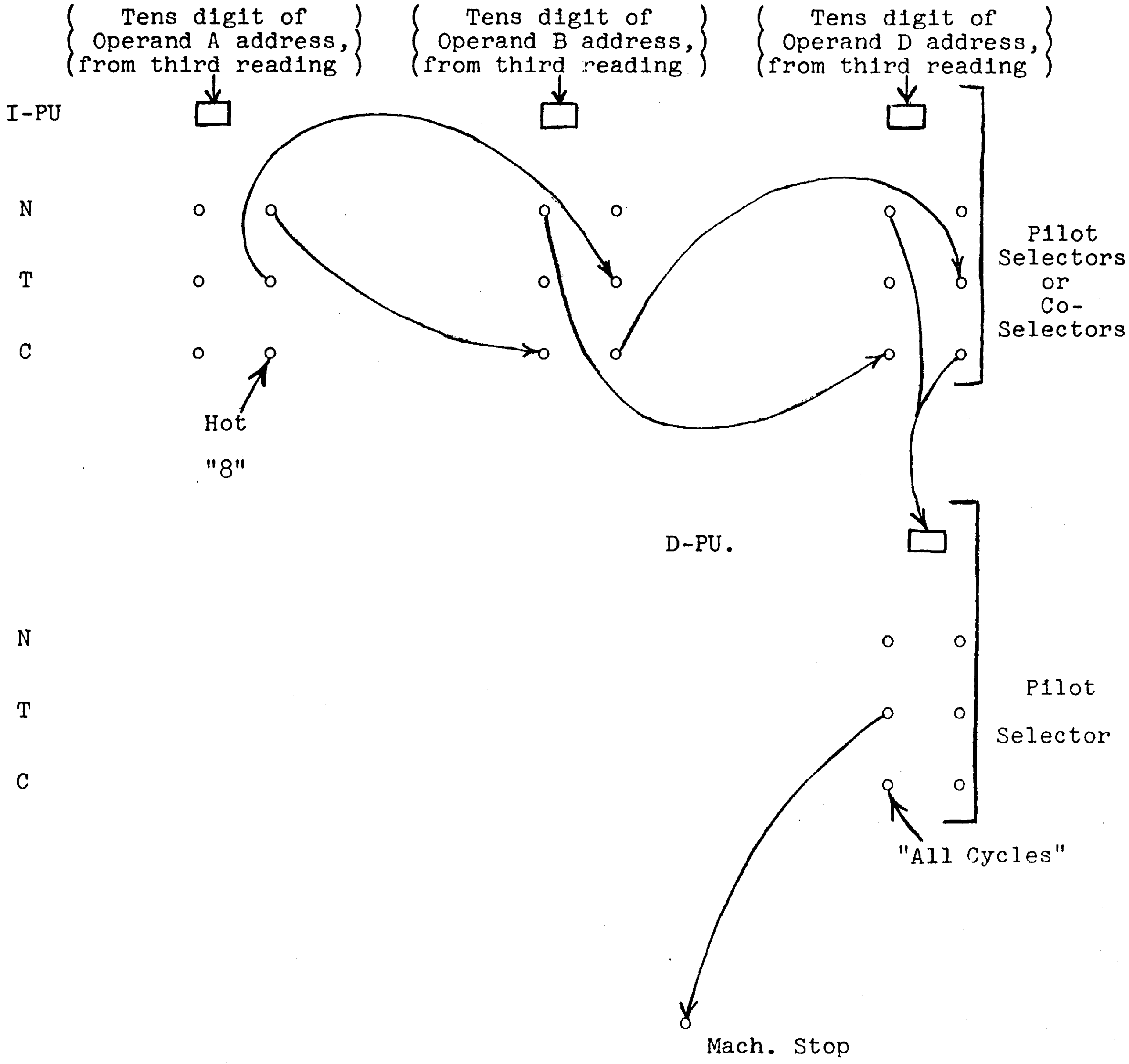
To clear the 605, the start button must be depressed for several cycles before the cards are placed in the feed hopper. It may happen that the zero check light (sometimes also the unfinished program light) will turn on during these clearing cycles. If so, then the reset button must be depressed following each cycle.

The 605 panel wiring is given in Figure 2, and the 527 panel wiring in Figure 3. Figure 4 shows the flow chart for the 605 panel wiring.

*Note: If an error is detected near the end of a deck, so that the card-feed hopper empties and causes feeding to stop before the error card is ejected into the stacker, then the start button should be pressed until the zero check light causes feeding to stop, with the error card the last one ejected in the stacker.

This eventuality may be avoided, if desired, merely by placing two blank cards at the end of the deck of instruction cards.

Figure 1


Tens digit of Operand A address, from third reading
Tens digit of Operand B address, from third reading
Tens digit of Operand D address, from third reading
I-PU
N
T
C
Hot "8"
Pilot Selectors or Co-Selectors
D-PU.
N
T
C
Pilot Selector
"All Cycles"
Mach. Stop

ACCOUNTING MACHINE WIRING TO CHECK

FOR TYPE 1-a ERRORS

Figure 2

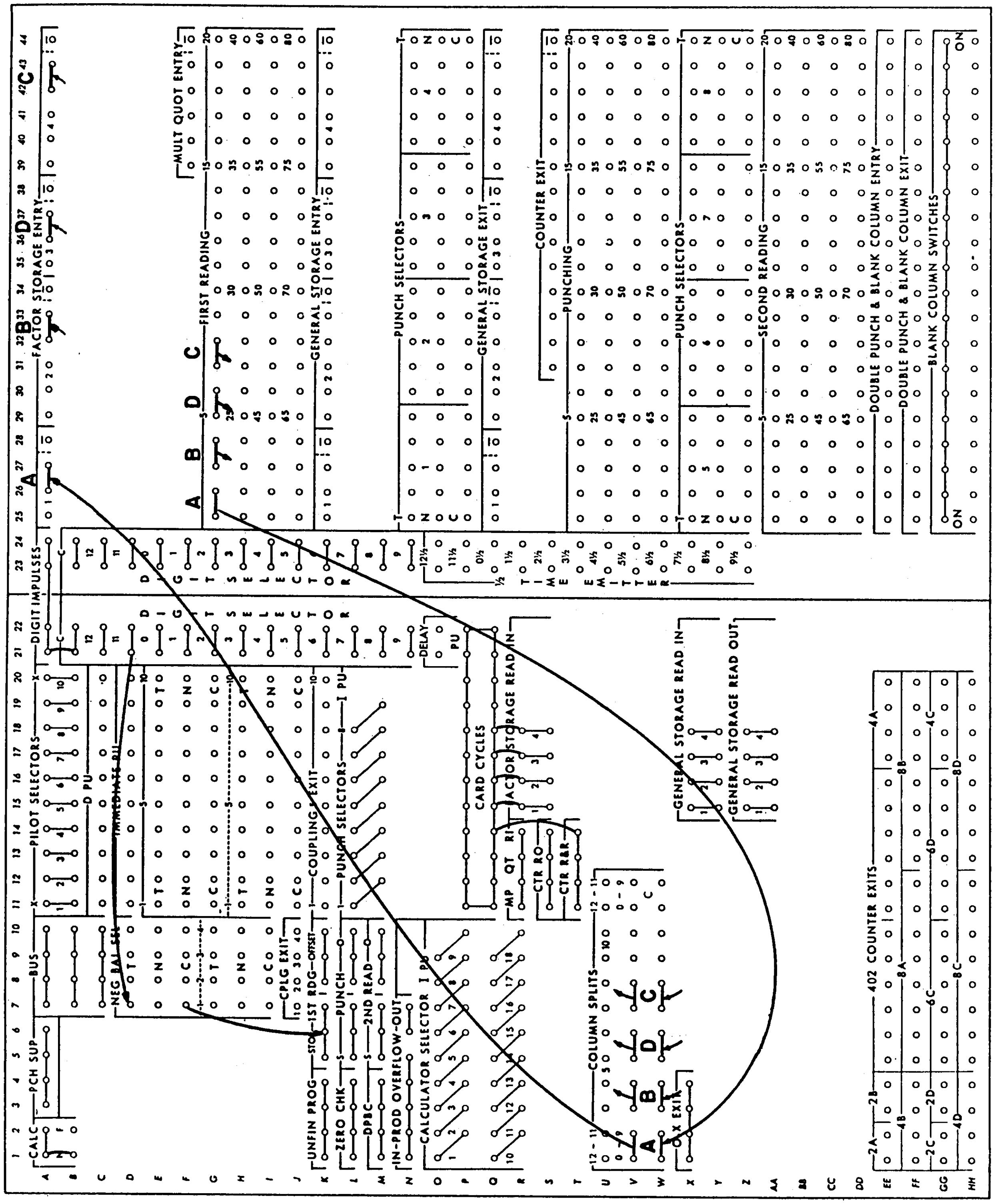
STEP	SUPPRESSION		FUNCTIONS		
			1.	2.	3.
1			Emit 8	CRI -	Into 2
2			F4RO	CRI +	B.T.
3			Reset		
4	(-)		F1RO	CRI +	
5	(-)		F4RO	CRI -	
6	(-)		Z.T.	Reset	
7	(-)	N.Z.		(P.U. GR.	SUPP 1)
8	(-)		F2RO	CRI +	
9	(-)		F4RO	CRI -	
10	(-)		Z.T.	Reset	
11	(-)	N.Z.		(P.U. GR.	SUPP 1)
12	(-)		F3RO	CRI +	
13	(-)		F4RO	CRI -	
14	(-)		Z.T.	Reset	
15	(-)	N.Z.		(P.U. GR.	SUPP 1)
16			F1RO	CRI +	
17			G2RO	CRI -	
18			Z.T.	Reset	
19	N.Z.			(P.U. GR.	SUPP 2)
20			F2RO	CRI +	
21			G2RO	CRI -	
22			Z.T.	Reset	
23	N.Z.			(P.U. GR.	SUPP 2)
24			F3RO	CRI +	
25			G2RO	CRI -	
26			Z.T.	Reset	
27	N.Z.			(P.U. GR.	SUPP 2)
28			G2RO	CRI +	
29			Z.T.	Reset	
30	N.Z.			(D.O. GR.	SUPP 2)
31			G1RO	CRI +	
32			Z.T.	Reset	
33	N.Z.			(P.U. GR.	SUPP 4)
34	(4)		F1RO	CRI +	
35	(4)		G1RO	CRI -	
36	(4)		Z.T.	Reset	
37	(4)	N.Z.		(P.U. GR.	SUPP 1)
38	(4)		F2RO	CRI +	
39	(4)		G1RO	CRI -	
40	(4)		Z.T.	Reset	
41	(4)	N.Z.		(P.U. GR.	SUPP 1)
42	(4)		F3RO	CRI +	
43	(4)		G1RO	CRI -	
44	(4)		Z.T.	Reset	
45	(4)	N.Z.		(P.U. GR.	SUPP 1)
46	(4)		G1RO	CRI +	
47	(4)		F4RO	CRI -	
48	(4)		Z.T.	Reset	
49	(4)	N.Z.		(P.U. GR.	SUPP 1)
50			G2RO	CRI +	
51			Emit 8	CRI -	Into 2
52			B.T.		
53	(-)			(D.O. GR.	SUPP 2)
54			G1RO	G2RI	
55			F4RO	G1RI	Reset
56			Emit 3	CRI -	
57	(1)	(2)	Emit 5	CRI +	
58			(B.T. FOR SEL. P.U.)		

605 PANEL WIRING FOR

TIMING CHECK PROCEDURE

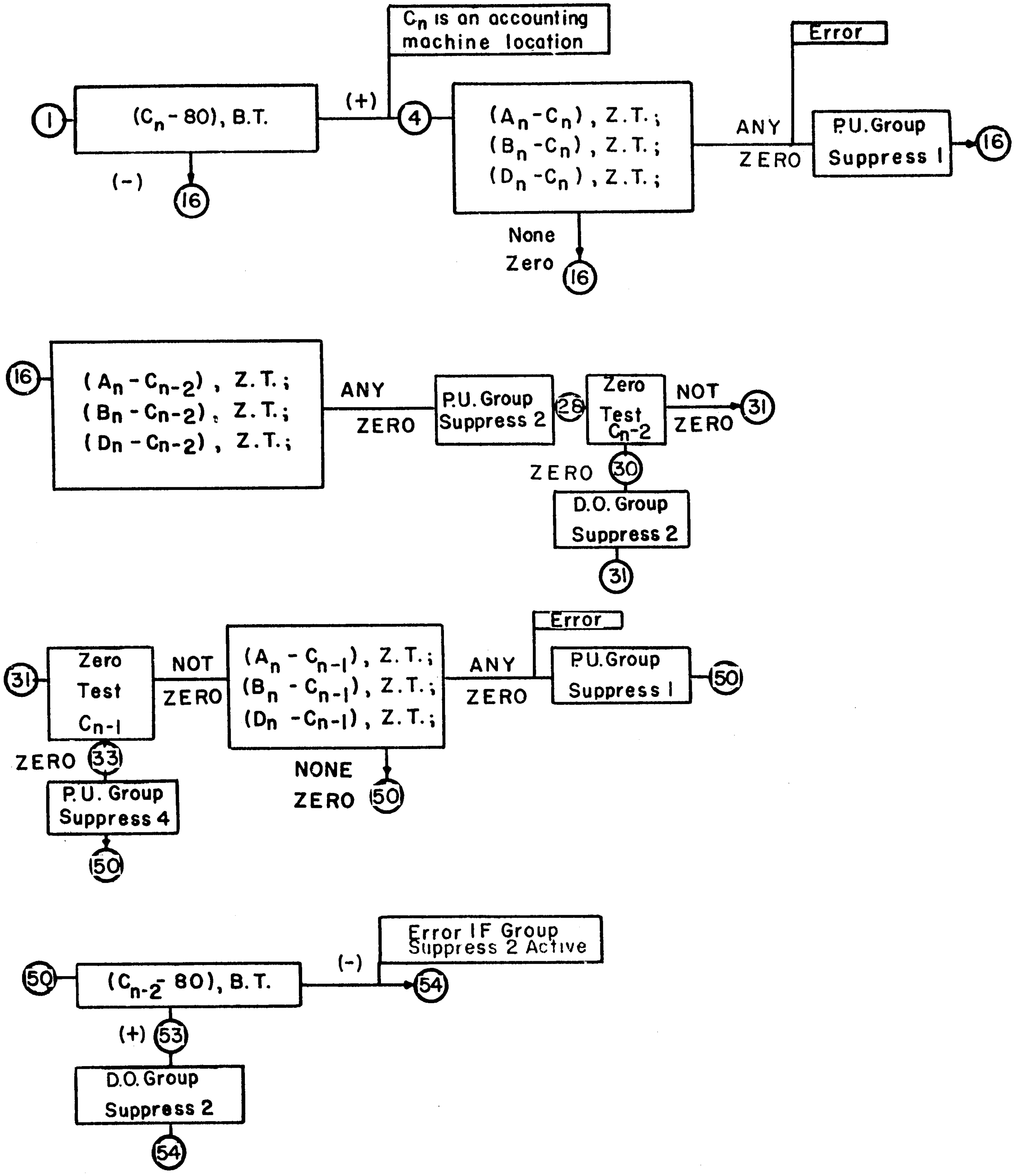
Figure 3

TYPE 521 PUNCH UNIT



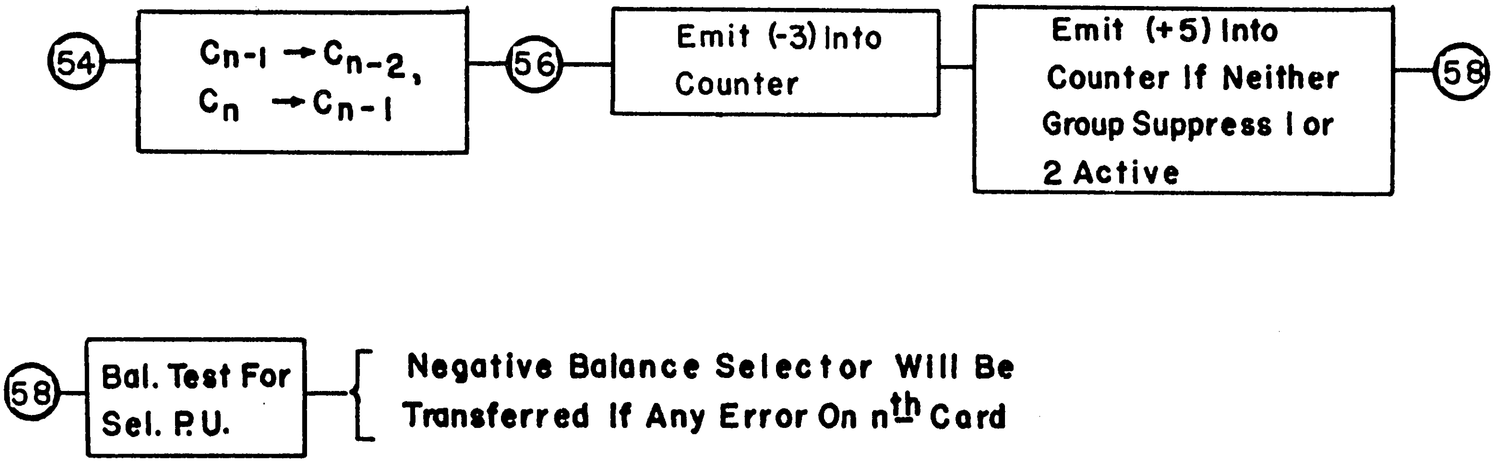
527 PANEL WIRING FOR TIMING CHECK PROCEDURE

Figure 4a



FLOW CHART - TIMING CHECK PANELS

Figure 4b


54
$C_{n-1} \rightarrow C_{n-2}$,
$C_n \rightarrow C_{n-1}$
56
Emit (-3) Into Counter
Emit (+5) Into Counter If Neither Group Suppress I or 2 Active
58
58
Bal. Test For Sel. P.U.
Negative Balance Selector Will Be Transferred If Any Error On nth Card

LEGEND:

A_n, B_n, D_n ARE ADDRESSES OF THE THREE OPERANDS ON n^{th} CARD

C_n IS ADDRESS FOR THE RESULT ON THE n^{th} CARD

B.T. = BALANCE TEST FOR STEP SUPPRESSION

Z.T. = ZERO TEST

P.U. = PICK UP

D.O. = DROP OUT

CIRCLED NUMBERS ARE KEYS TO THE CORRESPONDING PROGRAM STEPS ON FIGURE 2.

References:

1. Hamming, R. W., "A 101 Control Panel for Testing General Purpose CPC Cards," Technical Newsletter No. 9, Applied Science Division, IBM Corporation, January, 1955.

2. Orchard-Hays, William, "The Duplex System for IBM's Model II CPC," The Rand Corporation, Report RM-1044, February, 1953.

ERRATA

Technical Newsletter No. 8

Page 13: Paragraph 4 should read: At a card punching rate of 100 cards per minute, up to 554 ms are available for computing; and at a card reading speed of 200 cards per minute, up to 254 ms are available for computing. (The interlock timing for the 533 is 46 ms for both read and punch.)

Page 22: It has been discovered that the following locations are not used by the floating decimal interpretative system: 0047, 0173, 0319, 0353, 0372, and 0423. Locations 0019-0024 are not used either, but they were purposely left blank in order to allow the addition of 6 more operations if so desired. Location 0466 is used by the system, however, making a total of 467 memory locations required.

Page 23: General interpretation, line 12 should read:

```
0040   SU    11    0193    8003
```

Page 24: Problem description at the top of the page should read:

$(A) + (K) \longrightarrow K$ Subroutine
$(K) = a_1 A_1$, $(A) = a_2 A_2$, $a = xx$, $A = x.xxxxxxx\pm$

Page 25: In the floating decimal multiplication subroutine, instruction 0181 will result in an overflow in the case of all 9's. This will in turn invalidate the two subsequent Store Address instructions.

Page 29: Interpretation of 07, lines 14 and 15 should read:

```
0243   RAL   65    B       0465
0465   STL   20    0089    0134
```

Page 31: Interpretation of 11, line 2 should be replaced by the following 2 instructions:

```
0417   RAL   65    8002    0425
0425   AU    10    0037    0378
```

Page 31: Interpretation of 13,lines 1 and 2 should be replaced by the following 3 instructions:

```
0013   SRT   30    0002    0271
0271   RAL   65    8002    0229
0229   AU    10    0037    0431
```

Page 34: Interpretation of 16, line 35 should read:

```
0326   2100000026
```

Page 36: Interpretation of 18, lines 1-4 should be replaced by the following 3 instructions:

```
0018   SRT   30    0002    0384
0384   STL   20    0389    0442
0442   RAL   65    0029    0401
```

Page 40: First line should read: Store next instruction in 0083.

Page 41: In the Remarks column, $a_3 = a_1 = a_2 -1$ should be replaced by $a_3 = a_2 - a_1 - 1$.

Page 61: It has been discovered that location 0065 is not used by the double precision arithmetic program.

ERRATA

Technical Newsletter No. 9

Page 27: Line 9 should read: A control word of the form $n \ n^2 + A_0 \ \ A_0$.

Page 34: $1.\theta_1 = 0$ for the sine subroutine should read: $1.\theta_1 = \theta$ for the sine subroutine.

Page 35: It has been discovered that location 0077 is not used by the sine-cosine subroutine SC1.

Page 37: Line 3, opposite instruction location 0006 should read: Compute $\theta_1 = \pi/2 - \theta$. If $\theta_1 \geqq 0$, subtract 2π from it. Note: For the cosine subroutines, it is necessary to place $-\theta$ in the upper half of the accumulator.

Page 51: Paragraph 2, lines 1-2 should read: This subroutine is initiated by placing the argument x = x.xxxxxxxxx in the upper accumulator.

It has been discovered that location 0023 is not used by the logarithm subroutine.

The Matrix-Vector multiplication sub-routine requires that all factors be less than 1 in absolute value. Thus a_{ij}, x_j = .xxxxxxxxxx

The following table lists the required contents of the accumulator when entering the various sub-routines and the results left in it upon their completion.

Sub-Routine		Upper	Lower
SR1	Entry	0	A
	Exit	0	$\sqrt{A}$
SR2		Can be anything	A
		0	$\sqrt{A}$
SC1, SC2		θ	0
		0	sine θ or cos θ
Arcsin		x	Can be anything
		arcsin x	intermediate results
Arctan		x	0
		arctan x	intermediate results
Arctan by Radix Table		0	x
		arctan x	intermediate results
e^x		x	Can be anything
		0	e^x
1n (1+x)		x	0
		1n (1+x)	intermediate results

IBM®

INTERNATIONAL BUSINESS MACHINES CORP.
590 MADISON AVENUE, NEW YORK 22, N. Y.

Litho. in U.S.A.