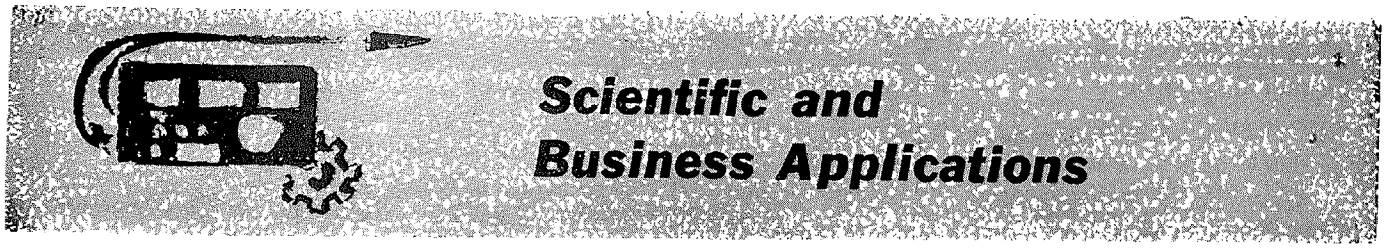


D. T. TEICHROEW, Editor

A Computer User-Oriented System

GEORGE D. MONTILLON
The Procter & Gamble Company, Cincinnati, Ohio

A computer language system has been developed which makes possible fast preparation of management reports, regardless of computational complexity or format variety. Costs are sufficiently low so that individually tailored reports can be prepared for every manager.

The system requires initial preparation of large data banks containing data in elementary form. Use of two special languages, EXTRACT and MATRAN, permits selective extraction of any data subset, efficient processing through any computational sequence, and flexible presentation of results in either tabular or graphical form. Matrix algebra is used as a fundamental vehicle for accomplishing both manipulation and computation.

1. Introduction

The problem we set out to solve about four years ago can be characterized by three phrases: masses of data, complex analyses and urgency. The highly competitive nature of our business dictates that we consider an ever increasing amount of data. Studies of the package detergent market, for example, might involve data for over 100 brand sizes, over many time periods, in many geographical areas. It is no longer unusual to start a study with more than 100,000 numbers. To describe any real situation accurately requires use of a large number of characteristics or variables with complex relationships. A relatively small number of them, fortunately, may provide an adequate description. Our ability to analyze the situation falls off rapidly as we increase the number of variables used and as we depart from simple assumptions about their relationships.

The net result of this dilemma is that we approach a result by successive approximation. Studies are frequently characterized by a preliminary stage of data reduction followed by manager or analyst decision to determine the direction of further study. Successive stages may involve repetition of a given analytical approach at a different level of detail, or they may involve completely different analytical approaches. Many stages may become necessary, each dependent upon all prior stages. Timeliness is essential in our work; quick use of latest available data in appropriate analyses is necessary for effective management. Our solution to this problem involved extensive use of fast computers, development of flexible programming languages, and generation of modular operating programs to implement analysis and reporting.

More important than the fact that we used computers extensively is the *manner* in which we coordinated the work of our managers and the operation of our computers. The basic problem is to make use of the combined strengths of manager and computer to the fullest extent possible. The problem is difficult because the work of the manager is not susceptible to detailed description, accurate measurement, incisive analysis or graphical display. We concluded more than two years ago that our best contribution could be made, not by defining better analytical methods as such, but rather by increasing the opportunity for experimentation with a variety of analytical methods aimed at clarifying the highly complex and continually changing problems of our markets. Such experimentation, we believe, should be carried out *jointly* by a line manager and a staff analyst, using the computer for as much of the implementation work as possible. The resulting mutual education of both manager and analyst should lead to better analytical approaches than either one could devise by himself, especially if limited to manual methods.

The strengths of the managers, both line and staff, lie primarily in intangible areas widely discussed but little understood in quantitative terms. Typical descriptive terms would be innovation, intuition, judgment, adaptability, imagination, generalization and experience. The strengths of a computer, by way of contrast, lie primarily in clearly definable areas, such as speed, accuracy, consistency and low cost of computation. Even those who are intimately involved with the development of computer technology remain largely unaware of the economic impact of rapidly changing unit costs of computation. During the past fifteen years, we have witnessed the entire development of the high-speed electronic computer. Every five years, the cost of a given computational job has been cut to one-tenth of its previous cost. Considering further equipment developments already in process, we can expect this exponential decrease to continue for several more years. This means that many system uses of the computer,

Presented at the 3rd Annual Technical Symposium sponsored by the San Francisco Bay Area Chapter of the ACM, San Francisco, California, November 1, 1963.

which appear completely unjustified today, will be economical by the time the systems are implemented.

2. System Requirements

Traditional systems analysis implies that system requirements should be defined in *detail* before initial design takes place. We chose, instead, to define only in very general terms our requirements for creation of data banks or files, selective extraction capability, analytical programming capability and report programming capability.

Creation of data banks, or files. Since any set of data was potentially usable for many different studies, we stored available data in raw form for greatest flexibility. Even if current studies required only summary data, we built data banks in as basic form as we could, anticipating that future studies would require finer units or more complex classification. Consider a shipment file comprising three two-way classifications of the basic data: brand sales by quarters for the total of all districts, district sales by quarters for the total of all brands, and brand sales by districts for the total of several quarters. Such summary files preclude the later use of finer units, such as sizes within a brand, sales units within a district, or months within a quarter. Furthermore, they preclude the use of any three-way classification of a subset, such as brand sales by quarters for a selected set of districts. On the other hand, a single large file of shipments in a three-way classification, with finer units in each dimension, makes any summary feasible.

Selective extraction capability. Although it is possible to define the kinds of data required to solve as-yet-undefined problems, it is not practicable to define the formats in which such data would be useful. Both file creation and selective extraction have common characteristics. Both require recognition of elements regardless of format and manipulation of elements into new formats. Therefore, one language, EXTRACT, was developed to implement these requirements. This language provides complete independence from format restrictions.

Analytical programming capability. Because of the unpredictable nature of our problems and the evolutionary process required for their solution, we established several subsidiary requirements. First, programs had to be usable independently, to preclude running through a long program in order to use a small part of it. Programs written as small modules, or building blocks, however, could be made as efficient as was found necessary or desirable. Such programs could remain unchanged regardless of changes in other programs; they could be revised immediately, on the other hand, if their own efficiency could be increased. Second, programs had to be linkable in any conceivable order, with any conceivable amount of repetition, sequential or intermittent. Such programs would be somewhat inefficient because of the compatibility requirements at junction points, but they would provide unlimited flexibility. Third, retention of intermediate results in both tape and print form had to be feasible. Such retention is desirable for several reasons: if a program should fail in a long sequence of computations, intermediate results could be used to restart at the point of failure; if the analytical approach required modification near the end of a sequence, earlier computations would not have to be repeated.

Report programming capability. In the past, emphasis on computing costs has often resulted in computer printouts which have been difficult for both analysts and managers to read and understand, because they have included inadequate descriptions, confusing decimal point conventions, superfluous digits and inflexible formats. Consequently, much time and effort was required to transcribe numbers from computer printouts and construct appropriate tables and charts for management attention. Computer production of *readable* reports, therefore, was considered essential. These reports had to be self-explanatory, with format flexibility adequate to meet the requirements of individual managers. Further, they had to be usable directly in reports, without transcription and subsequent error.

Most managers depend primarily upon tabular presentations of information. Many managers, on the other hand, can perceive relationships more easily when information is presented in graphical form. Computer production of graphs, therefore, had to be feasible in conjunction with other computer work.

Both analytical and report programming also have common characteristics. Both require manipulation of sets of elements, either numeric or alphabetic. Both require ability to make changes rapidly, either in mathematical procedures or physical arrangements of elements. Therefore, a second language, MATRAN, was developed to implement these requirements. This language provides complete flexibility in both manipulation and computation on element sets.

3. Languages

The data banks with which we operate comprise files of records, each of which combines one or more attributes with one or more pieces of numerical or alphanumerical data. Typical attributes include time periods, geographical locations and product descriptions. Typical data include shipments, consumption, marketing activity, distances, costs and population.

GENERAL FILE RECORD FORMAT

ATTRIBUTE 1	ATTRIBUTE 2	...	ATTRIBUTE M	DATUM 1	DATUM 2	...	DATUM N

TYPICAL FILE RECORD FORMAT

BRAND	SIZE	DISTRICT	YEAR	MONTH	QUANTITY

EXTRACT. Regardless of file format, we must be able to extract whatever data is of interest on the basis of certain attributes. The EXTRACT language accomplishes

such extraction through a three-stage process. The first

EXTRACT

DESCRIBE INPUT FORMAT	SPECIFY SELECT CONDITIONS	DEFINE OUTPUT FORMAT

stage describes the input format completely as to its length and contents. A file record for shipments, for example might include the following fields: BRAND, SIZE, DISTRICT, YEAR, MONTH and QUANTITY. The second stage specifies the conditions, either basic or compound, under which a given record is to be extracted.

Basic condition statements define a logical relationship between one input field value and a constant or another input field value. The permissible relationships include EQUAL, LOWER THAN, HIGHER THAN, ZERO, MINUS and PLUS; the negatives of these; and BLANK. Each basic condition must have a unique name. SELTIDE, meaning select Tide, would be represented by the statement, BRAND E TIDE. Similarly, SELSIZE 60 and SEL1962 would be represented by SIZE E 60 and YEAR E 62, respectively. Lists of included items within a given attribute may be used in a single statement, as in SELT&C, where BRAND E TIDE (&) CHEER. SELLARGE might describe quantities not lower than 10,000; SELNOTNEG might describe quantities not minus; SELONEMAX might describe quantities not higher than 1.

Compound condition statements define logical connections between or among the truth values associated with the names of either basic or compound conditions. The permissible connectives include the asterisk, representing the logical AND; the ampersand, representing the logical OR; and the minus sign, representing the logical NOT; in any sequence, AND takes precedence. Each compound condition must also have a unique name. SELT&C might also describe the statement, SELTIDE & SELCHEER. SELTN60&C, meaning select Tide, except size 60, or Cheer, might describe the statement, SELTIDE*-SELSIZE60 & SELCHEER. SELZT01, meaning select within the range from zero to one, might describe the statement SELNOTNEG*SELONEMAX.

Either basic or compound conditions may be used to specify extraction; any logical condition may be generated by combining basic and compound conditions. All names are completely arbitrary; they are usually selected for their mnemonic value, and may include any combination of up to 10 alphanumeric characters. The third stage defines the desired output format for the extracted records. This format can include all or any part of the record, rearranged in any desired way; for example, the output records might contain only the brand, sales district and number of shipments. In any case, the original data bank is always preserved. The primary use for EXTRACT is data retrieval, but it has also been found useful for many additional purposes. Editing of data for completeness and consistency within individual records is an important secondary use. Making unusual or one time changes to many records within a file is another important use. Most important, perhaps, is the ability to make major file format changes. This can be done incidental to extraction for some other purpose; the additional cost is then negligible.

MAFORM. EXTRACT lacks two capabilities which we need to provide suitable input to MATRAN. It processes individual records only, so that sorting by desired row and column attributes is subsequently necessary; it includes no arithmetic capability, so preliminary batching of data is not feasible. We have introduced a linking program, called MAFORM for matrix formation, to overcome these problems.

EXTRACT OUTPUT FOR MAFORM

MATRIX NAME	ROW SEQ	ROW NAME	COLUMN SEQ.	COLUMN NAME	QUANTITY

TYPICAL MAFORM OUTPUTS

DISTRICTS	MONTHS
	1 SIZE
	1 DISTRICT
	1 MONTH

YEARS	BRANDS
	ALL SIZES
	ALL DISTRICTS
	12 MONTHS

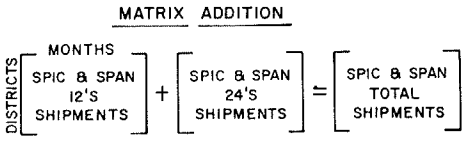
Operating on the typical file record discussed earlier, EXTRACT can produce a specific type of record suitable for MAFORM. This record includes an identifying name for the matrix in which the record will be included, row and column names for use in titling, row and column sequence numbers if alphabetical order is not desired, and the quantity. A typical output matrix might include districts as row and months as columns. The individual data element might represent shipments for one size in one district for one month. Another typical output matrix, involving preliminary summarization, might include years as rows and brands as columns. The individual data element in this case might represent shipments of all sizes of each brand in all districts over twelve-month periods. In addition to data matrices, MAFORM provides vectors of row and column titles for use in preparing final reports.

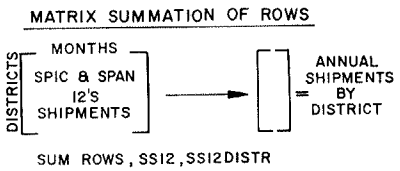
MATRAN. The basic logistics problems of large-scale business analysis had long since emphasized to us the need for powerful analytical tools. Matrix algebra offered the most immediate gains, both for simple and complex manipulations. In the longer run, further, it promises to become the primary conceptual tool of the analyst.

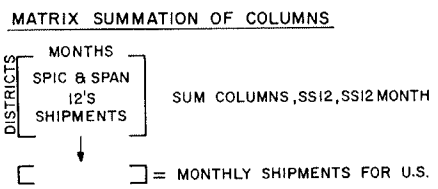
MATRAN, meaning Matrix Algebra TRANslator, was originally conceived of as primarily a mathematical language. Its subsequent evolution has made it a very powerful general-purpose language. MATRAN is convenient for even the simplest of arithmetic operations. One instruction is sufficient to perform any number of like operations, assuming the operations apply to all elements of compatible matrices or vectors. For example, assume we want to determine total Spic & Span shipments for all districts and for all months of a year, given the corresponding data

for the two sizes, 12's and 24's. The MATRAN instruction would be simply: ADD, SS 12, SS 24, SS TOTAL.

The operands, defining which matrices are being operated upon and where the result is to be stored, are arbitrary names. Again, assume we want to determine annual shipments by district for Spic & Span 12's. The appropriate MATRAN instruction is: SUM ROWS, SS 12, SS 12 DISTR. Similarly, if we want to determine monthly shipments for all districts, the MATRAN instruction is: SUM COLUMNS, SS 12, SS 12 MONTH. In such operations as these, the primary advantage of MATRAN is the elimination of work for the user, since each MATRAN instruction replaces a double-nested loop.







The advantages of MATRAN are much more obvious when the problems are more complex. Consider the problem of multiple regression, a statistical tool now in daily use (Figure 1). Here we see the problem and its solution in matrix notation:

$$Y = XB$$
$$B = (X'X)^{-1}X'Y$$

To program this solution in MATRAN, we must first define necessary storage areas, which requires six names and statements. Next, we must carry out the mathematical steps in appropriate order, which requires only nine statements, or a total of fifteen. This compares with several thousand instructions required in Basic Autocoder. Equivalent operations in FORTRAN would require about 20 instructions, assuming existence of INVERT and MULTIPLY subroutines.

In order to illustrate the general versatility of MATRAN, it is necessary to describe some of the operational details. MATRAN definitions are of four kinds: matrices, vectors, scalars and integers. Each definition must be identified by a unique name, comprising up to ten alphanumeric characters. Each matrix must have two dimensions, representing maximum numbers of rows and columns; each vector must have one dimension, representing maximum number of elements. MATRAN operands include the names of these matrices, vectors, scalars and integers, as well as the names of statements. All of these sets of elements are represented internally as matrices. Each matrix is represented by a row dimension, a column dimension and a set of elements equal in number to the product of row and column dimensions. A scalar or an integer is simply a 1×1 matrix; elements, unless integers, are in floating-point form.

MATRAN incorporates a variety of arithmetic operations. Element-by-element operations are the most widely used. Operations with scalars form another useful class. Summation of elements is easily done, either in total, or by rows or columns. The most powerful arithmetic operations involve matrix multiplication and inversion. Compound operations such as TRANSPOSE MULTIPLY and DIAGONAL VECTOR MULTIPLY offer efficiency both in programming and in storage requirements. Several special arithmetic operations are also available, including selection and identification of minima and maxima.

MATRAN incorporates the usual kinds of transformation operations. These are accomplished element-by-element, and the transform replaces the original element.

Movement operations are equally as important as arithmetic operations in providing versatility. MATRAN provides two general kinds of movement. MOVE transfers any matrix, vector, scalar or integer to another location at high speed. TRANSPOSE interchanges corresponding elements of rows and columns in a matrix or vector.

Operation on individual elements of a vector or vectors of a matrix requires that these components be isolated. MATRAN provides for extraction of single elements, of a row or a column of any matrix or vector, and of the diagonal of a square matrix. After operation on the isolated components is completed, they must usually be replaced in their previous positions or inserted in some new position.

MULTIPLE REGRESSION

PROBLEM — $Y = X B$

SOLUTION — $B = (X'X)^{-1} X'Y$

DEFINE STORAGE —

```
X      MATRIX , 100 , 30
XPX    MATRIX ,  30 , 30
D      SCALAR
Y      VECTOR , 100
XPY    VECTOR ,  30
B      VECTOR ,  30
```

COMPUTE —

```
READ CARDS , X
TRANSPOSE MULTIPLY , X , XPX
INVERT , XPX , D
READ CARDS , Y
TRANSPOSE , X
MULTIPLY , X , Y , XPY
MULTIPLY , XPX , XPY , B
PRINT , B
END RUN
```

FIG. 1.

The replacement operations in MATRAN move elements or vectors without modifying dimensions. The INSERT VECTOR operation changes appropriate dimensions and moves all affected vectors or elements of the receiving matrix or vector to accommodate the addition.

Sequence control is a vital part of any general-purpose language. MATRAN provides three main types of sequence control. Transfers are provided for in usual ways. The GO TO instruction provides for unconditional transfer to the statement named as operand. IF provides for transfer to appropriate statements depending upon whether the first operand is negative, zero or positive. COMPARE provides for transfer to appropriate statements depending upon whether the second operand is lower than, equal to, or higher than the first operand in the sort sequence. END RUN provides for transfer to the next MATRAN program.

Repetitive execution of one or more operations, or looping, is provided for by use of the START LOOP and END LOOP operations. These two statements are logically identical to the FORTRAN DO statement. START LOOP must have a name to which END LOOP refers. Operands of START LOOP define an indexing integer, its starting and ending values, and the indexing interval.

Especially important for our purposes is the provision for including subroutines. The PERFORM operation starts at the ENTRY statement, continues through the EXIT statement, then returns to the statement following PERFORM. Any number of subroutines may be used, and subroutines may be nested (i.e., one subroutine may call another as long as the second does not in turn call the first). By use of this device, we have been able to generate metalanguages of various kinds to serve special purposes.

Another important characteristic for analytical purposes is dimension control. Dimensions of one matrix or vector are frequently needed to control an operation or to establish a compatible matrix or vector elsewhere. Dimensions can be extracted with ROW TO INTEGER or COLUMN TO INTEGER operations, or replaced by the reverse operations. DIMENSION establishes the row dimension of the first operand and the column dimension of the second operand as the two dimensions of the third operand.

Several self-explanatory types of operations are available for element-by-element conversions. FIX, FLOAT, and CLEAR operate only on numeric data; BLANK and SHIFT operate on alphanumeric data.

The several MATRAN printing operations make it possible to produce finished management reports directly from the computer. EDIT converts floating-point numbers into fixed point with desired number of decimal places. PRINT ALPHA provides for printing alphanumeric data with desired spaces between columns and rows. MASK provides for superimposing characters from one element upon another. COMMENT and MESSAGE provide for introduction of editorial notes into a program, or into the output of a program, respectively. SPACE and RESTORE assist in improving legibility.

Finally, MATRAN provides a variety of input-output options. Cards for numeric data may be read or punched all at once for a given matrix or one row at a time. READ PUNCH CARD implements reading of alphanumeric data. Printing of numeric data can also be done all at once for a given matrix or one row at a time. Elements are in unedited form suitable for analysis but not for finished reports. Standard tape operations complete the major MATRAN statements. Inherent in the operations described is a high level of flexibility and convenience to the user.

Characteristics of the Language. These computer languages, in summary, have very desirable characteristics.

COMPUTER LANGUAGES
USED IN
MANAGEMENT INFORMATION SYSTEMS

1. EASY TO LEARN	2 MAN - DAYS
2. EASY TO WRITE	USE ENGLISH WORDS USE FEW INSTRUCTIONS
3. EASY TO CHECK	PIN POINT ERRORS
4. INEXPENSIVE	
TO PROGRAM	1 - 8 MAN-HOURS
TO RUN	$2 MORE / PROGRAM

In the first place, they are easy to learn; formal training of two days is usually sufficient for either EXTRACT or MATRAN. Prior understanding of matrix algebra is desirable but not critical, and minimal knowledge about machines is required. These languages are easy to write, because they use English words and mnemonic names; relatively few instructions are required. Although a separate instruction is required for each logical phrase of an equation, the time required to write and check is not markedly greater than with FORTRAN.

These languages are easy to check since the language translators developed by Data Processing Research have built-in diagnostic routines, and a few minutes inspection of a program failure usually locates an error. These languages are inexpensive to program, since most programs require less than a full man-day of effort. The resulting load-and-go programs are inexpensive to run; inefficiencies resulting from modularity are balanced by efficiencies of matrix subroutines. The assembly cost of about two dollars per program is a strength rather than a weakness; reassembly at each use permits continual revision, both of the language itself and of the programs. We seldom run a program twice without revision; vested interest in existing programs is minimal; it is frequently easier to write a new program than to adapt an existing program for modified use.

4. Applications

Applications of this approach have been made in many areas. Some applications are of a repetitive nature; others occur intermittently; still others are one-shot analyses.

Repetitive applications. One class of repetitive reports involves shipment summaries for Advertising Department Brand Managers. These reports include geographical subdivisions as rows. Most include all districts and divisions; many include special subsets of districts, selected by area or by physical characteristics. The columns of these reports include individual months, subsets of months, indices of recent to prior periods and rankings among the districts.

SHIPMENT REPORTS

TYPICAL ROWS	TYPICAL COLUMNS
ALL DISTRICTS	INDIVIDUAL MONTHS
ALL DIVISIONS	SUB-SETS OF MONTHS—
SUB-SETS OF DISTRICTS—	LAST X MONTHS
BY AREAS	PREVIOUS Y MONTHS
BY CHARACTERISTICS	Z MONTHS YEAR AGO
	INDICES
	RANKINGS

The important distinction between these reports and conventional shipment reports is that these are tailored to the individual desires of the Brand Managers involved. Standardization for its own sake is eliminated. Standardization for programmer convenience is eliminated. Standardization occurs only if the managers involved decide that it aids mutual understanding of operating problems.

Another class of repetitive reports involves delivery expense summaries for Traffic and Sales Department managers. Here both rows and columns are more diverse because the reports are used by more than one operating function.

In addition to geographical subdivisions, carriers, order size classes and customers appear as rows on some of the reports. The columns include such headings as number of orders, number of cases, weight, actual freight cost, cost per hundred weight, cost per case, reference or ideal freight cost, and indices of performance.

DELIVERY EXPENSE REPORTS

TYPICAL ROWS	TYPICAL COLUMNS
ALL DISTRICTS	NO. OF ORDERS
ALL DIVISIONS	NO. OF CASES
SUB-SETS OF DISTRICTS	WEIGHT
CARRIERS	ACTUAL FREIGHT COST
ORDER SIZE CLASSES	COST PER CWT.
CUSTOMERS	COST PER CASE
	REFERENCE FREIGHT COST
	INDICES

Whereas the shipment reports dealt with monthly summaries, these reports deal with individual orders.

As with the shipment reports, these reports are tailored to the desires of the managers involved. Reports for traffic managers are distinctly different from reports for sales managers.

These two classes of reports have much in common. They are generated primarily from listings of the row and column titles desired. Consider, for example, the column title, LAST 6 MONTHS. This would be written in normal fashion by the user. It would be key-punched as separate 10-character elements, left-justified for convenience of the key-puncher. The computer would then scan the elements, store the "6" for subsequent use, detect and right-justify the longest word, center other words or numbers relative to the longest, and generate appropriate underlining. The computer would then prepare for printing by stacking the words and centering over the edited number field.

COLUMN TITLE SPACING

```
WRITE —
    LAST 6 MONTHS

KEY PUNCH —
 |LAST      |6          |MONTHS

COMPUTE
 |     LAST |      6    |     MONTHS|      ------|

PRINT —
          LAST
            6
         MONTHS
         ------
         XX.X±*
           .
           .
         XX.X±*
```

A subroutine would calculate the appropriate sum of the last 6 months after deriving the last month from the basic data matrix dimensions and the first month from the "6" set aside above.

TYPICAL SUBROUTINE

1. DERIVATION OF CONTROLS

```
COLUMN TO INTEGER , BASICDATA , LASTMONTH
EXTRACT ELEMENT, 1 , 2 , COLTITLES, DURATION
SUBTRACT, LASTMONTH , DURATION , FIRSTMONTH
ADD , 1. , FIRSTMONTH , FIRSTMONTH
```

2. HOUSEKEEPING & DESCRIPTION

```
PERFORM , COLSETSUM , COLSETEND
   .
   .
COLSETSUM DIMENSION , BASICDATA , 1 , SUMVEC
          CLEAR , SUMVEC
          COMMENT, SUM A SET OF CONSECUTIVE COLUMNS
          COMMENT, FROM FIRSTMONTH THROUGH LASTMONTH
          COMMENT, AND PLACE IN COLUMN 1 OF REPORTDATA
          .
          ( COMPUTATION INSTRUCTIONS )
          .
COLSETEND RETURN
```

3. COMPUTATION INSTRUCTIONS

```
COLSUMLOOP START LOOP, J , FIRSTMONTH , LASTMONTH , 1
           EXTRACT COLUMN , J , BASICDATA, STOREVEC
           ADD , STOREVEC , SUMVEC , SUMVEC
           END LOOP , COLSUM LOOP
           REPLACE VECTOR , SUMVEC , 1 , REPORTDATA
```

This typical subroutine would make use of the PERFORM operation, with arbitrary names for entry and exit. COMMENT might be used to describe the subroutine in detail for the convenience of the users. This typical subroutine would also make use of a variable loop, since it should handle the summing of any set of consecutive columns. The two controls, FIRSTMONTH and LASTMONTH are sufficient.

Similar logic applies to derivation of controls from such titles as PREVIOUS Y MONTHS and Z MONTHS

YEAR AGO. PREVIOUS must be interpreted relative to LAST X MONTHS, however.

Intermittent Analyses. Facility location represents an important class of intermittent analyses. Estimated tariffs are appropriate for many such studies. In such instances, geographical location coordinates for the cities being considered as sources or destinations are convenient for calculating straight-line inter-city distances. Application of appropriate tariff equations and addition of source-specific costs yields a total unit cost matrix. Multiplication by the demand vector then yields total cost to fulfill any included destination's demand from any one of the included sources.

The total cost matrix can then be used to evaluate any selected subset of sources. After selection of the best source in the subset for each destination, summaries of costs and allocated demands are easily made. This approach costs so little that many possibilities can be evaluated and the sensitivity of total cost to various strategies can be examined in depth.

MAJOR STEPS IN
FACILITY LOCATION ANALYSIS

FOR EACH POTENTIAL SOURCE OR DESTINATION —

EXTRACTION OF LOCATION COORDINATES
CALCULATION OF DISTANCE MATRIX
APPLICATION OF TARIFF EQUATIONS
ADDITION OF SOURCE-SPECIFIC COSTS
MULTIPLICATION BY DEMAND VECTOR

TO YIELD TOTAL COST
° FROM ANY INCLUDED SOURCE
° TO ANY INCLUDED DESTINATION

FOR EACH SELECTED SUB-SET OF SOURCES —

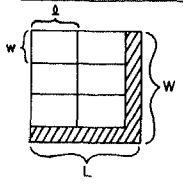
SELECTION OF BEST SOURCE FOR EACH DESTINATION
LISTING OF DESTINATIONS SUPPLIED BY EACH SOURCE
SUMMATION OF DEMAND SUPPLIED FROM EACH SOURCE
SUMMATION OF COSTS INCURRED AT EACH SOURCE
SUMMATION OF ALL COSTS

One-Shot Analyses. Typical of one-shot analyses carried out with these techniques is pallet capacity analysis. The problem is simply to determine best arrangements of packages on pallets for various sizes of pallets and packages. This problem has been attacked usually by drafting techniques.

PALLET CAPACITY ANALYSIS

FOR ANY GIVEN PALLET SIZE
FOR EACH GIVEN PACKAGE SIZE
FOR EACH OF TWO PATTERN CLASSES
SELECT MOST EFFICIENT PATTERN

PARALLEL PATTERNS

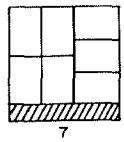


L = Pallet Length
W = Pallet Width
ℓ = Package Length
w = Package Width

The most obvious kind of pattern involves placing the packages in parallel rows. This type of pattern is a subset of what we might call perpendicular patterns. All of the possible perpendicular patterns are described by the function within brackets, where the partial brackets represent the largest integer not exceeding the enclosed quantity (Iverson's concept of the "floor"). Evaluation of this function for all permissible values of I and J is quite inexpensive for realistic values of the basic dimensions. The number of possible patterns for a given package did not exceed 64 in the actual runs.

PALLET CAPACITY ANALYSIS

PERPENDICULAR PATTERNS



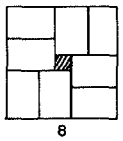
$I = 0,1,\cdots,\left\lfloor \frac{L}{w} \right\rfloor$

$J = 0,1,\cdots,\left\lfloor \frac{W}{w} \right\rfloor$

$F(I,J) = \max_{I,J}\left[I\left\lfloor \frac{W-Jw}{\ell} \right\rfloor + J\left\lfloor \frac{L}{\ell} \right\rfloor + \left\lfloor \frac{L-Iw}{\ell} \right\rfloor \left\lfloor \frac{W-Jw}{w} \right\rfloor \right]$

There is another kind of pattern, however, which may turn out to be more efficient than the perpendicular patterns. These chimney stack or pinwheel patterns require a considerably different treatment. This project is of particular interest, because it illustrates several important points. The problem required a major contribution from an analyst. It was solved by computer enumeration of small finite subsets within certain algebraic restrictions. Results of both kinds of analysis were presented to the user for final decision about such factors as operating stability. The division of labor among user, analyst and computer seems appropriate.

CHIMNEY STACK PATTERNS



$I = 1,2,\cdots,\left\lfloor \frac{L-\ell}{w} \right\rfloor$

$J = 1,2,\cdots,\left\lfloor \frac{W-\ell}{w} \right\rfloor$

$G(I,J) = \max_{I,J}\left[(1+J)\left(\left\lfloor \frac{L-Iw}{\ell} \right\rfloor + \left\lfloor \frac{W-Jw}{\ell} \right\rfloor \right) \right]$

s.t. $G(I,J) \leq \left\lfloor \frac{LW}{\ell w} \right\rfloor$

The applications I have cited emphasize tabular presentation of results. Graphical presentations can also be incorporated if the user desires. Graphing routines have been found useful in preliminary steps to analysis, as well as in summarizing analytical results.

5. Summary

The original stimulus for this development came from the Management Systems Section of the Advertising Department early in 1961. Accomplishments to date are attributable to close collaboration between the Data Processing Systems Research Group, the Industrial Engineering Systems Analysis Department, and Management Systems-Advertising. D.P.S. Research developed the necessary languages and I.E.D. Systems Analysis developed the appropriate programs to implement the operating requirements defined by Management Systems-Advertising.

This approach is quite similar to that developed under the auspices of the Minuteman High Reliability Component Program as described by Wang [1]. It is somewhat

more general, but probably less efficient, than the AUTOSTAT language described by Douglas and Mitchell [2].

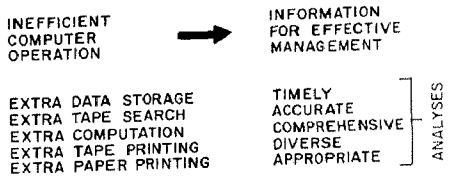
Our approach is based on the premise that the primary role of a computer in management information systems work is *not* to provide present information at lower cost, but rather to provide appropriate *new* information never before available to the manager. This new information should include those analyses which the staff analyst and the line manager jointly decide are needed, based on exercise of their unique but intangible special strengths. Frequently their decision will be dependent upon one or several previous analyses, from which they have gained additional insight about their problem area. The new information desired may represent a new direction of analysis or more detailed treatment in a previous direction.

What have we really accomplished through this approach? One major gain is reduction of the analytical and managerial time required for a given problem. Back in 1961 a typical problem required

1 week to define the problem
4 weeks to collect the data
9 weeks to analyze the data
2 weeks to present conclusions

for a total of 16 weeks. That elapsed time has been cut to a total of four weeks today. Although the definition time remains the same, the time required to collect the data, analyze the data, and present conclusions have each been reduced to about one week. Data collection is faster, because most of the pertinent data is already available on tape files. Analysis is much faster, because of language simplicity and program modularity. Even presentation of conclusions is somewhat faster, because most appendices are produced directly as computer output, then photoreduced to standard report size.

What kinds of systems have resulted from this emphasis? At first glance, they seem to ignore every ordinary criterion of computer operating efficiency. We provide for *extra data storage*, because its marginal cost is small. We plan on *extra tape searching* to provide flexibility. Elapsed managerial time is reduced at the expense of computer running time. We plan on *extra computation*, because the marginal cost of computation is small compared with input and output costs. We plan on *extra tape printing* and *paper printing* to permit exercise of managerial judgment at many points in an analysis.



Have we really gained from this emphasis? Our experience confirms that we have. We have been able to provide more timely, more accurate and more comprehensive analyses. Diverse approaches have yielded more appropriate final results. A single recent project has paid for the total development cost of this approach several times over. In all these cases, we are trading computer operating inefficiencies for better management information. Fortunately, these added costs have turned out to be a small percentage of our total computing costs. Yet these apparent inefficiencies provide us with tremendous flexibility. The *primary* advantage of such flexibility lies in our ability to *defer decisions* until the most appropriate time.

We can start developing data banks without having standardized either format or content; we can modify both format and content whenever it is desired. We can start analyses without having decided how to complete them, change direction easily, perform in parallel many steps which would normally require sequential treatment, and we can examine more alternatives within given time limits. We can incorporate latest data quickly, defer action until maximum information is available, then act swiftly in response to peculiar market conditions. We can provide much detail initially, then reduce detail or supplant it with exception reports, as users gain experience with the system.

The primary disadvantage of our approach relative to FORTRAN and COBOL is the need to write a new compiler whenever we change to noncompatible machines. This cost is estimated to be about one man-year. There are several primary advantages over FORTRAN and COBOL.

1. Matrix algebra is a more natural language for our analysts in the context of most business problems than ordinary algebra.
2. MATRAN statements, though more numerous when handling nonarray data are generally simpler.
3. Input-output is considerably more convenient.
4. Compiling speed of 1000 statements per minute permits limitless modification at negligible cost.

The low cost of compilation itself compensates severalfold for the cost of rewriting the compiler every few years.

What have we learned during the development of this approach? If we were to start over at this point, we would probably modify the capabilities of our languages in several respects. EXTRACT might be designed to handle more powerful logic, including parenthesized statements, as well as simple arithmetic operations. Ability to make interrecord comparisons would be very helpful. MATRAN might be designed for greater flexibility in definition. Names of matrices and titles of rows and columns could supplement the present identification by location and dimensions. In general, however, we would probably follow the same evolutionary approach. We have developed a workable system, starting with a *minimum* of assumptions about the problems to be solved. Further development of the system is not hindered by our past decisions.

RECEIVED NOVEMBER, 1963; REVISED OCTOBER, 1964

REFERENCES

1. WANG, T. L. An information system with the ability to extract intelligence from data. *Comm. ACM 5* (Jan. 1962), 16–18.
2. DOUGLAS, A. S., AND MITCHELL, A. J. AUTOSTAT: a language for statistical data processing. *Comput. J. 3* (July 1960), 61–66.