CLM-R 45

CLM-R 45

United Kingdom Atomic Energy Authority

RESEARCH GROUP

Report

SCIENTIFIC COMPUTING AND OPERATIONAL RESEARCH

K. V. ROBERTS

Culham Laboratory,

Culham, Abingdon, Berkshire

1965

Available from H. M. Stationery Office

FOUR SHILLINGS AND SIXPENCE NET

U.D.C.
681.3

UNCLASSIFIED
(Approved for Sale)

CLM - R 45

SCIENTIFIC COMPUTING AND OPERATIONAL RESEARCH

by

K.V. ROBERTS

A B S T R A C T

This report describes operational research carried out at the Culham Laboratory, designed to improve the efficiency of both scientific computing and system programming by the introduction of automatic techniques.

U.K.A.E.A. Research Group,
Culham Laboratory,
Nr. Abingdon,
Berks.

July, 1965 (ED)

C O N T E N T S

1. INTRODUCTION

The development of computer hardware and software has now reached a critical, almost explosive stage. In some respects the improvement in performance during the last one or two years represents several orders of magnitude; for example on-line computing allows the man-machine interaction time to be reduced from the usual 1-24 hours to 1-5 seconds. A direct consequence of this has been an enormous extension of the range of scientific problems for which computers are potentially useful.

Some of the developments in hardware are illustrated in Table 1. Main core storage and random-access backing storage have both become much larger and cheaper. A wide range of visual output is available, and well adapted to scientific use: microfilm (including ciné film), photographic hardcopy, and the on-line cathode-ray tube. The computer can accept input from a light pen, and read microfilm. Some on-line typewriters cost little more than desk calculators, and can be attached to standard telephone lines. The recently-announced CDC 6800 and IBM 360/95 have instruction speeds several tens of times faster than that of the IBM 7090, which until not long ago was the standard machine for scientific use. At the other end of the scale there are small machines such as the PDP 8 which cost only a few thousand pounds, and can be attached to individual experiments.

If some of the claims put forward for advanced computing techniques sound exaggerated to many scientists, it is largely because people have become adjusted to the considerable inconvenience of even the best existing installations. Until recently, unless a scientist wished to carry out a large number of similar calculations as in bubble-chamber analysis, or had a mathematical problem which could not be solved in any other way, he would have been well advised to avoid computers altogether, and to use analytic techniques or desk machines. As an illustration, we shall discuss in this report the elementary computing problem of tabulating and plotting an arbitrary mathematical function - in the simplest case, a standard function such as $J_0(x)$. It would take only a few minutes to look this up in a library; using a computer one must either write a Bessel function subroutine or extract a copy from a card or paper tape file, write a main program to read in and print out the data (working out an appropriate format), punch and handle cards or paper tape, fill in a job slip, compile and debug the program, waiting hours or sometimes days for each set of output to arrive. The results must then be plotted out by hand; alternatively if an automatic plotter is available it is usually necessary to program this in considerable detail. Quite clearly therefore, the computer does not yet begin to compare in efficiency with other well established information-handling systems such as libraries

and automatic telephones. For these systems, access to information has for many years been highly organised, by means of well-printed books, scientific journals, directories, catalogues, indexes, dialling codes, enquiry services, classification schemes and the like. The scientific computer user on the other hand has been forced to accept long delays and to do much routine mental and manual work for himself, a paradoxical situation in view of the fact that three main characteristics of computers are precisely their high speed, their ability to perform large numbers of routine manipulations without error, and the readiness with which all forms of automation may be introduced.

To resolve this situation it is essential to carry out a thorough programme of operational research into the whole process of scientific computing, questioning every procedure and piece of equipment that is currently used, determining the specific causes of delays, errors and routine human effort, and eliminating them so far as possible. This problem has been recognised in the United States, where there is now considerable emphasis on improving the man-machine interaction[1], actively sponsored by bodies such as the Advanced Projects Research Agency of the U.S. Department of Defense. One example is Project MAC at M.I.T.[2,3,4] which enables a number of users to have on-line time-shared access to a central machine via individual typewriters, connected to the telephone network. Another is the Thompson-Ramo-Woolridge On-Line Computing Center at Space Technology Laboratories[5,6], where visual display is used. A list of on-line time-shared systems at May 1965 is given in Table 2. Little comparable work has been done in Britain because the necessary computer facilities have not been available. A certain amount of operational research designed to increase the efficiency of scientific computing has however been pursued at the Culham Laboratory, and some of the work carried out in the last three years will be described in this report.

2. BASIC 1401/STRETCH SYSTEM

Until recently* all work done by the Laboratory used the IBM 7030 (STRETCH) computer at AWRE Aldermaston, together with an IBM 1401 off-line or satellite machine situated at Culham itself[7]. Access was available to STRETCH two or three times each day according to a regular schedule, magnetic tapes being taken to and fro by taxi over a distance of about thirty miles. The AWRE STRETCH is a large configuration with 96K words of core store, 2 million words of disc storage, 16 magnetic tape units and other peripherals.

---

* July 1965. An English Electric-Leo-Marconi KDF9 was installed at Culham in April 1965 and most of the computing work of the Laboratory has now been transferred to this machine.

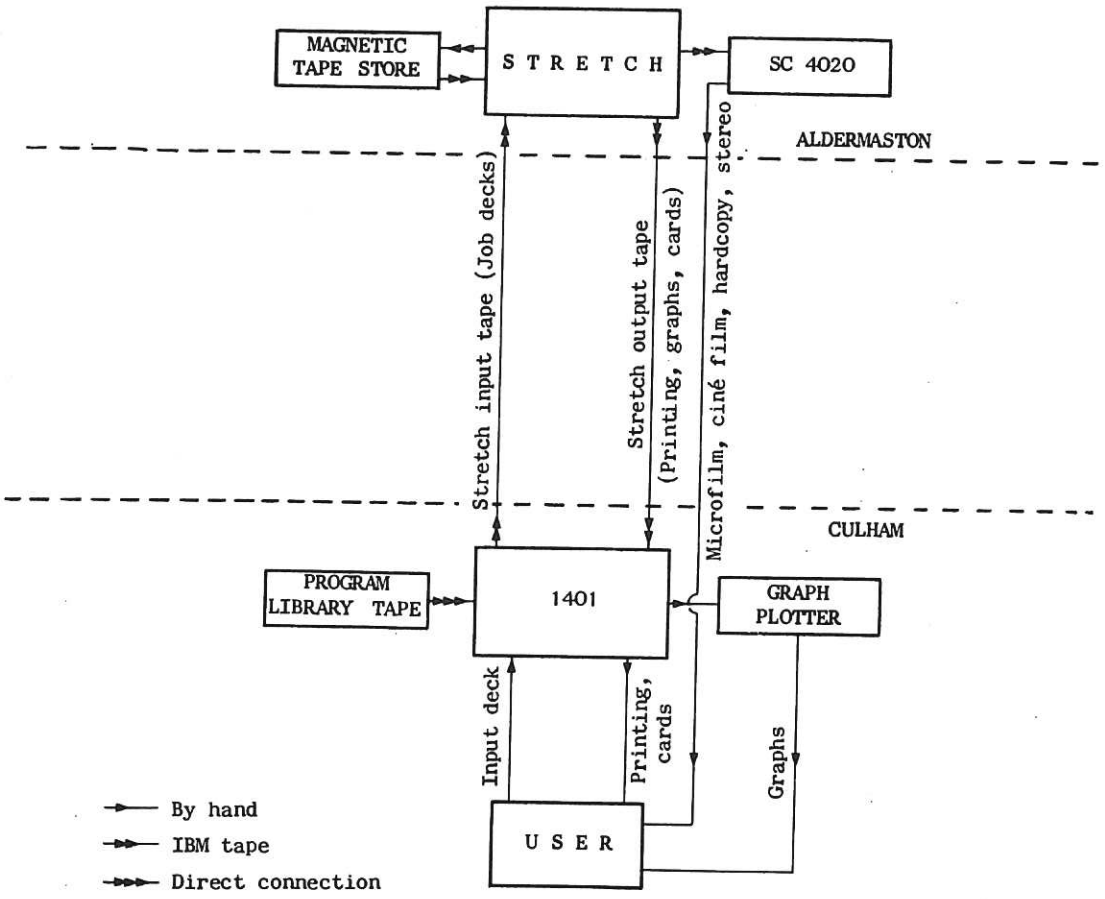


Fig.1 The Culham 1401 / STRETCH system (CLM-R45)

In addition there is a General Dynamics SC 4020 microfilm and photographic hardcopy recorder at AWRE and a Benson-Lehner magnetic-tape-controlled Model J graph plotter at Culham. All these facilities form part of an integrated system (Fig.1), so that simply by placing the appropriate control cards or FORTRAN statements in his input deck the Culham user can have access to any tape or to the disc, and can select his output in the form of printed tables, punched cards, graphs, diagrams drawn on the printer, microfilm, ciné film, stereographic slides or photographic hardcopy.

3. AUTOMATIC PROGRAM LIBRARY

In developing the Culham computer system, no attempt was made to alter the STRETCH software. This would indeed have been a formidable task, since MCP (Master Control Program) alone is said to have taken thirty-four man-years to write[8]. Effort was instead concentrated on building up a powerful library of FORTRAN subroutines and programs, and on improving the appearance of the system to the scientific user by modifying the 1401 input-output routines. The first major improvement, carried out by L.A.J. Verra in 1963, was to introduce an Automatic Program Library in order to eliminate unnecessary handling of cards.

As many computer users have realised, binary cards are anachronistic because once a deck has been punched out by the machine, nothing is ever done with that deck normally except to read it in again. Why then should it be punched out at all? In the past, the valid answer has been that random access backing storage was never large enough to handle all the subroutines compiled by a typical installation. Now that magnetic card files are becoming available, each capable of holding the equivalent of two million ordinary punched cards with a retrieval time of ⅓-½ second, it is likely that binary cards will disappear altogether, since one such file can store all the subroutines compiled by a major installation during more than a year.

No random-access device was available on the Culham 1401, but it was realised in 1963 that most of the cards used in production (as distinct from development) work for the Laboratory could be accommodated on one magnetic tape, and that the relatively long retrieval time would not constitute an unacceptable penalty.* A scheme was therefore adopted by which any sequence of cards in the user's Job Deck can be stored on a Program Library Tape (PLT) which is permanently mounted on the 1401, and referenced symbolically by means of a single L-card, (carrying 'L' in column 1, together with a File Number). Each L-card is the complete logical equivalent of the card file that it represents, when used in any job. Broadly speaking, all that has to be done† in order to implement this scheme is to modify the 1401 input routine (which copies cards on to the STRETCH input tape), so that it looks for the character 'L' in column 1. When this character is found, a number is read from the card and the corresponding file is copied from the PLT before proceeding to the next card (Fig.2).

This elementary modification allows the overall appearance of the system to the user to be radically simplified and generalised. For example, a complete program can now be represented by one card, even if it contains hundreds of separate subroutines, so that the entire job deck for a production run might comprise:

JOB-CARD
L-CARD
DATA

---

* This was because the 1401 was not fully utilised. For general use, it is recommended that the Program Library Tape should be replaced by some form of random access storage, e.g. an IBM 1311 Disc File.

† In fact the necessary modifications took one man about three evenings to make. This encourages the author's view that substantial improvements to system software can be made extremely quickly, provided that the existing version is properly understood, and all the ramifications of any change made clear. In this case, understanding was facilitated by the 'rigid' interface between the 1401 and STRETCH, beyond which no alteration could propagate. For further confirmation of this view, see also Section 11.

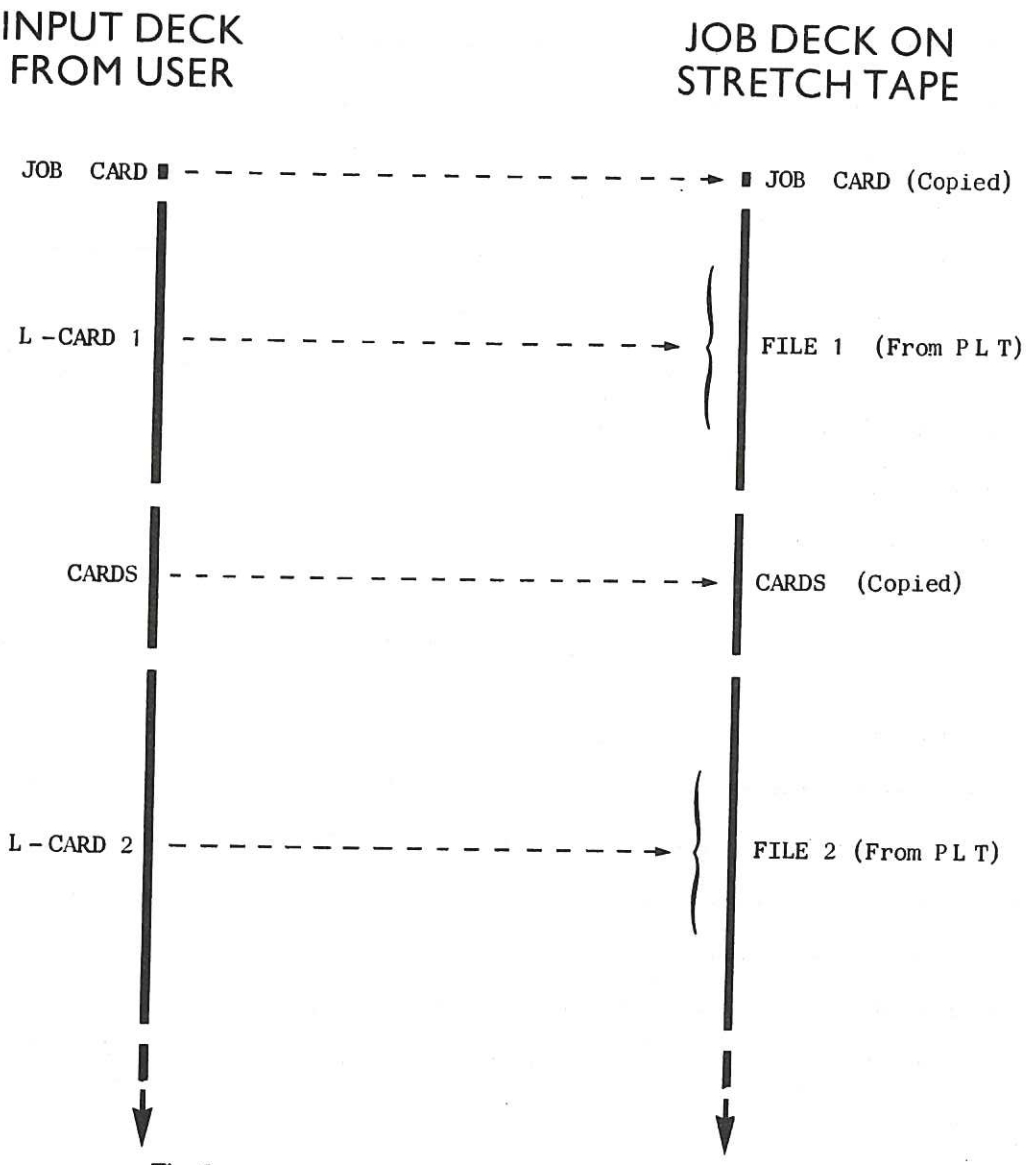


Fig. 2 (CLM-R45)
The L-card facility. Any sequence of cards in the Stretch Job deck (except the Job card itself) may be represented symbolically by a single L-card. This eliminates unnecessary card-handling, and the same L-cards may be used for public or private card indexes which can be printed or duplicated when required

Another useful device during the development of a large FORTRAN program is to put the current COMMON/DIMENSION deck on the PLT; one can then represent the set of declarations which must occur at the head of each subroutine in symbolic fashion by means of one L-card, which never has to be changed. This avoids the need for multiple copies of the deck, with all the consequent problems of updating as the list of declarations is increased.

4. CONTROL CARDS

A significant feature of card-oriented computer systems, which seems to give them a decided operational advantage over those using only paper tape, is the use of symbolic control cards, of which L-cards are one example. It is part of the Culham philosophy that

control cards are always prepunched, in order to avoid errors and to eliminate unnecessary human effort, and that preferably the stock should be kept up automatically. Copies are held in a tray near the reception desk and can be selected as required. The information on each card has previously been automatically interpreted by an I.C.T. 419 interpreter, (i.e. printed across the top), so that it can be read by the user as easily as by the machine. In addition to the character 'L' which initiates the substitution process, each L-card carries a File Number to indicate the position on the PLT, which is occasionally changed so that frequently used library items can be placed in the more accessible part of the tape. It also carries a permanent Reference Number* for the user, followed by a brief plain-language description of the purpose of the item.

5. SUBROUTINE LIBRARY

The subroutine concept is of immense significance in computing; it allows large programs to be built up from individual prefabricated 'bricks', thus greatly reducing the total amount of programming effort that is needed. FORTRAN systems have a further advantage in this respect, because it is possible to compile subroutines independently, into an intermediate machine-oriented language called Relocatable Binary (RLB). Furthermore if a single error is detected in the program, often only one subroutine has to be recompiled, which with STRETCH now takes only a second or so. Existing ALGOL systems usually require the whole of a large program to be recompiled afresh each time an error is found, and since compilation is in any case more difficult the relative penalty in machine time is extremely serious.

Each FORTRAN programmer usually divides his program into a number of private subroutines; here however we are concerned with the problem of organising a set of standard subroutines into a library. In our case this was greatly assisted by the fact that four separate A.E.A. computing groups used the same machine (Aldermaston, Culham, Harwell and Winfrith), and could therefore exchange programs freely. Most of the mathematical subroutines were in fact contributed by Harwell. Each subroutine is stored in RLB form on the PLT and represented by a single L-card. If a user wishes to employ a subroutine he need therefore only select the appropriate card from a tray and include it in his job; the subroutine will then be loaded automatically by the 1401.

---

* If a disc is used instead of the PLT, the Reference Number is sufficient and the File Number can be dispensed with, since the actual disc location is immaterial.

```
L RA0179 0055 *C SYMBOO   PRINTS ALPHABET ,GREEK ALPHABET AND NUMBERS ON PLOTTER
L  A0191 0427 *H OB03A    DRAWS LINE BETWEEN 2-PTS IN FREE RUN MODE
L  A0207 0376 *C PLVARO   PLOTS VALUE OF A VARIABLE ON BENSON-LEHNER
L  A0217 0734 *H OB04A    DRAWS SECTIONS OF THE SURFACE F(X,Y,Z)=0
L  A0218 0731 *H OB05A    PLOTS FORTRAN SYMBOLS ON GRAPH PLOTTER
                                                                        *
                                                                        *
 BESSEL FUNCTIONS
L   0021 0073 *H FF01A    COMPUTES BESS FUNCTS JO(X),YO(X) - REAL ARGS
L   0022 0074 *H FF02A    COMPUTES BESS FUNCTS J1(X),Y1(X) - REAL ARGS
L   0023 0075 *H FF03A    COMPUTES BESS FUNCTS IO(X),KO(X) - REAL ARGS
L   0024 0076 *H FF04A    COMPUTES BESS FUNCTS I1(X),K1(X) - REAL ARGS
L  A0196 0526 *H FF05A    EVALUATES SPHERICAL BESSEL FUNCTIONS
L RA0209 0375 *C BESSJ/KO BESSEL FUNTIONS OF COMPLEX ARGUMENT AND INTEGRAL ORDER
                                                                        *
                                                                        *
 BETA-FUNCTION
L R 0017 0069 *H  FC05A   COMPUTES BETA FUNCTION OF REAL ARGUMENTS
                                                                        *
                                                                        *
 CLOCK
L   0056 0036 *H ZAC1AS   READS STRETCH TIME INDICATOR
L   0178 0178 BM TMCHEQ    READS TIME REQD. IN MINS.,F5.2, AND REJECTS JOBS
                                                                        *
                                                                        *
 COMPLEX VARIABLE, FUNCTIONS OF
L   0075 0035 *C BESSJ/KO BESSEL FUNTIONS OF COMPLEX ARGUMENT AND INTEGRAL ORDER
L   0128 0128 *C CMPLXO   COMPLEX ALGEBRA
L   0142 0142 *C NEWTCO   MODIFIED NEWTON ITERATION TO A COMPLEX ROOT
LSR 0143 0143 *C          GRAPHICAL/ITERATIVE PROGRAM FOR COMPLEX ROOTS, CARD 1
LER 0144 0144 *C          GRAPHICAL/ITERATIVE PROGRAM FOR COMPLEX ROOTS, CARD 2
L   0145 0145 *C          GRAPHICAL/ITERATIVE PROGRAM FOR COMPLEX ROOTS, CARD 3
                                                                        *
                                                                        *
 CONICAL GUN
                SEE *THETA-PINCH GUN*
                                                                        *
```

Fig. 3 (CLM-R45)

Extract from the Classified Directory. This serves as an index to the facilities provided by the Culham system, and (apart from headings) is composed of the same symbolic L-cards that extract material from the library. The first number on each card is the reference number, which directs the reader to the relevant sheet in the Library Handbook (Fig. 4). Columns 1, 2-4 (for updating) and 10-13 (file number) are read by the machine. The symbols '*H, *C' denote Harwell or Culham subroutine respectively, while 'BM' refers to a private contributor. The next entry is the FORTRAN name of the subroutine. Updating is not always perfect; note that L75 should also appear under 'Bessel Functions'

L-cards in the tray are ordered according to Reference Number. When however copies of the same cards are placed in alphabetical subject order, together with suitable headings and cross-references, they can be listed on the printer to form a convenient Classified Directory to all the facilities available in the library, (rather like the Yellow Pages of an American telephone directory). Suppose, then, that a scientist wishes to use the Bessel function $J_0(x)$. He first looks up 'Bessel function' in his directory and finds that item 21 seems to provide the facilities needed (Fig.3). He then checks this by consulting Reference Sheet 21 in a loose-leaf Library Hand-Book (ordered according to reference number),

which gives full details of how to use the subroutine (Fig.4). Finally he places L-card 21 in his job.

Such a scheme can be extended indefinitely, and from an operational standpoint is simple both to use and to maintain. However, it still involves some degree of routine human effort, and on the KDF9 it is proposed that the Classified Directory should be stored on the disc and updated automatically whenever material in the main disk library is added, deleted or modified. Users can then always obtain a copy of the most up-to-date version by a "request" placed in the normal job stream.

As subroutine libraries become larger, it is clearly necessary that they should be catalogued as carefully as ordinary libraries containing books or reports, so that all the material is readily available to subscribers; however this standard is rarely met, in spite of the fact that the necessary automatic equipment is available.

Subroutine $FF_{01}A$ 21

1. Purpose

This subroutine calculates approximations to the Bessel functions

$$J_0(x) \quad \text{and} \quad Y_0(x)$$

The accuracy of the approximations is at least ten significant figures.

2. Argument List

SUBROUTINE $FF_{01}A$ (VJ_0, VY_0,X,N)

If N is set to 0 only $J_0(X)$ is calculated

If N is set to 1 only $Y_0(X)$ is calculated

If N is set to 2 both $J_0(X)$ and $Y_0(X)$ are calculated

The value of $J_0(X)$ will be set in VJ_0.

The value of $Y_0(X)$ will be set in VY_0.

3. Warning

If $J_1(X)$ is required X must be $\geqslant 0$.

If $Y_1(X)$ is required X must be > 0.

If X is negative $J_0(|X|)$ and/or $Y_0(|X|)$ are evaluated.

4. Method

A Chebyshev series is used if $0 \leqslant X \leqslant 8$. If X exceeds 8 a Chebyshev series in 1/X is used.

Fig. 4 (CLM-R45)
Sheet 21 of the Library Handbook, taken from the Harwell subroutine library

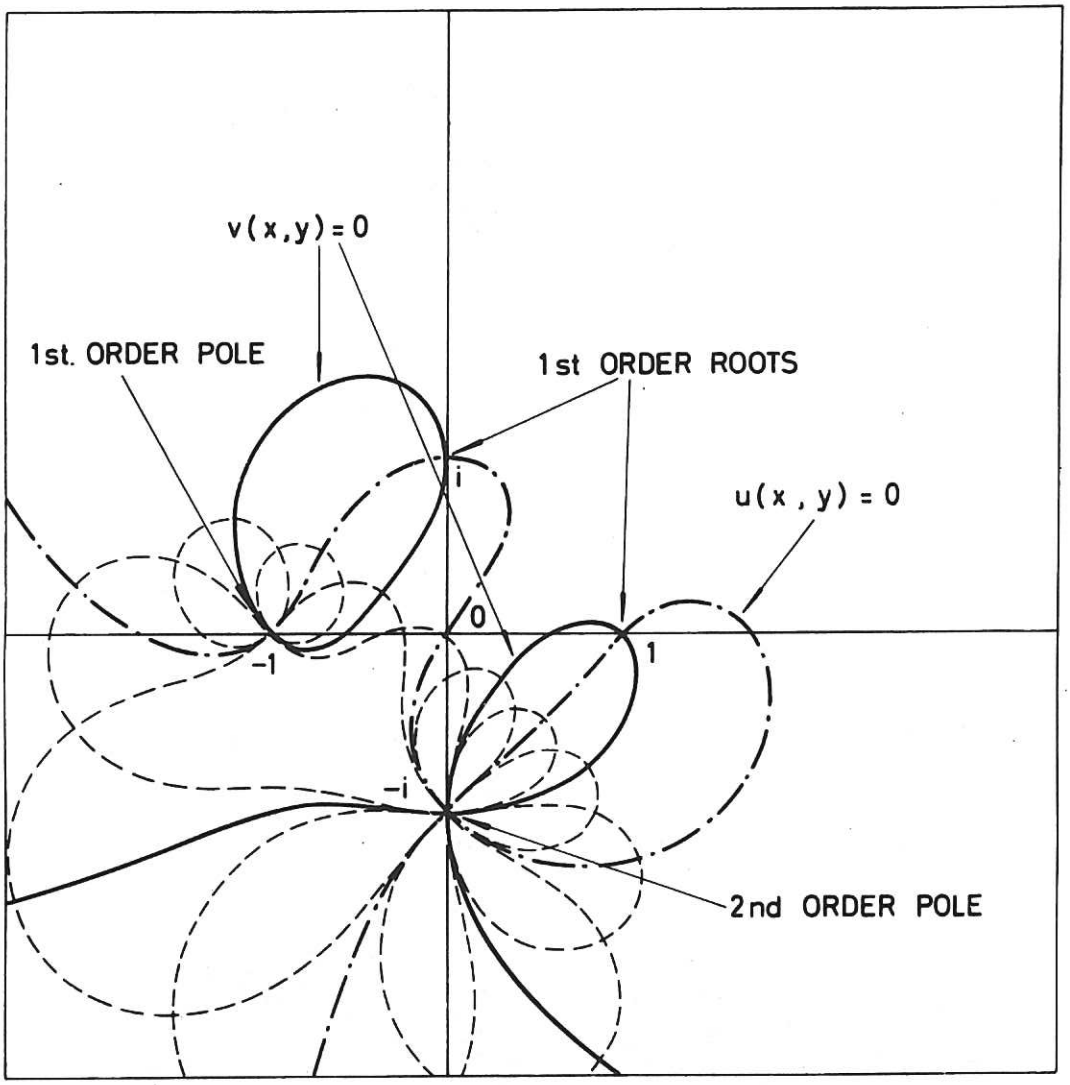


Fig. 5(a) (CLM-R45)
Contours of the real and imaginary parts of

$$f(z) = \frac{(z - 1)(z - i)}{(z + 1)(z + i)^2}$$

6. VISUAL OUTPUT

Graphs and diagrams of all kinds have always been useful in scientific work. The basic STRETCH system provides many types of visual output, but it would have been extremely wasteful in human effort for each individual programmer to make use of these facilities by starting from the fundamental plotter instructions afresh each time, and a comprehensive set of higher-level subroutines had to be provided. The first two of these were PLOT and KONTUR, written by F.M. Larkin in 1962, enabling points and curves to be plotted, and functions to be contoured. It is remarkable how many different applications can be found for contour plotting[9]; some examples are shown in Figs.5a and 5b. Since good scientific practice requires that descriptive material should be included on each graph, a lettering subroutine ALPHA was also provided.

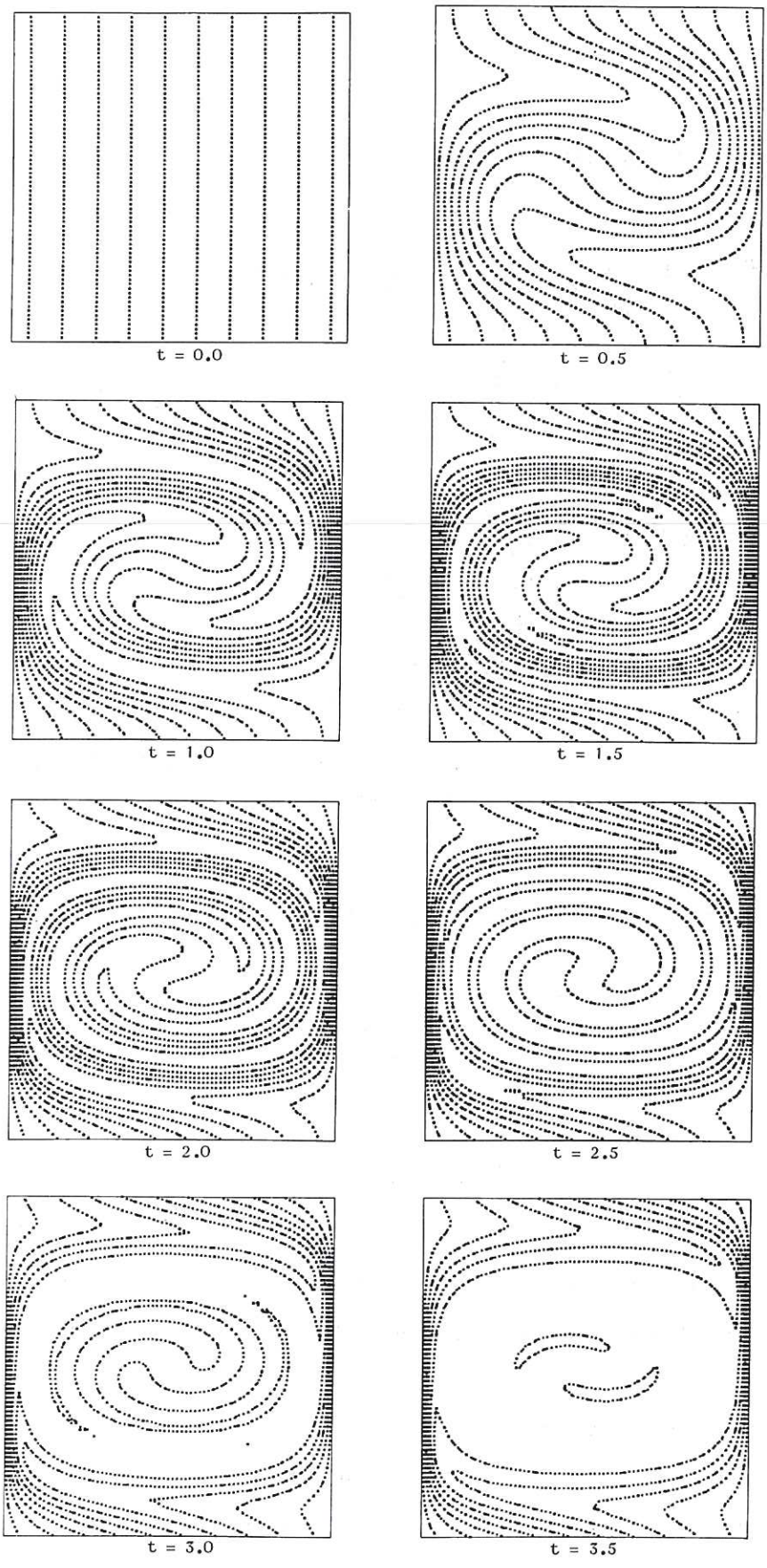


Fig. 5(b) (CLM-R 45)

Expulsion of an initially uniform magnetic field by an eddy in a partially conductive fluid. The lines of force are plotted as contours of the magnetic stream function
(Series of stills taken from a film by N. O. Weiss)

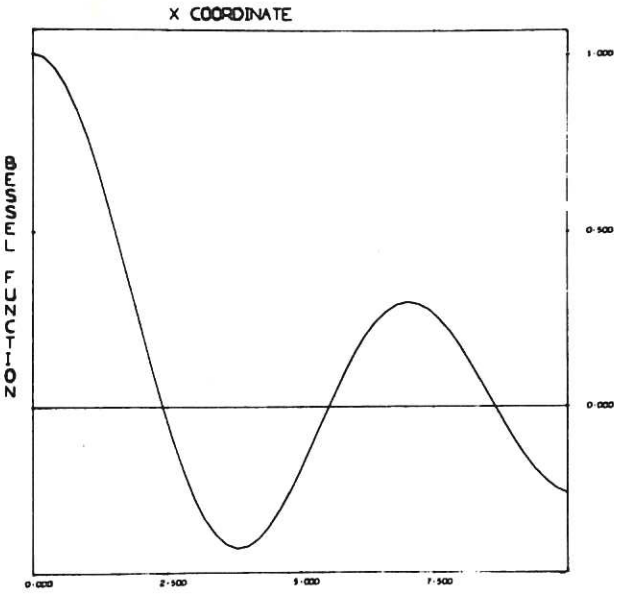


Fig. 6 (CLM-R 45)
SC 4020 output from Culham Standard Program 1, which plots and tabulates any continuous function automatically. Imperfections in the lettering are caused by the fact that the writing subroutine was originally provided for amechanical plotter, which 'cuts corners' due to inertia of the head. A better subroutine has now been provided for the SC 4020 (Fig.14). The users initials and date on card 14 of Fig. 8 normally appear in the upper left-hand corner, but have been omitted in the reproduction

Although contouring was now automatic, conditions were operationally still far from ideal, because in drawing an ordinary graph it was necessary to write instructions to turn the page and number it, insert the date, draw axes, work out and print scales, scale the function, plot successive points, and write the captions, all of which constituted a needless distraction from scientific work and ought certainly to be done by the machine. An attempt was therefore made by the author, (using a rudimentary form of time and motion study on himself), to find out how the work required from a programmer for plotting graphs might be reduced to an absolute minimum. A hierarchy of higher-level subroutines was written, all ultimately based on a version of PLOT, which would automatically scale and plot any function, reading captions punched on cards and writing them in the appropriate places (Fig.6). This set was extended by S.J. Roberts and N.O. Weiss to the point at which graph-plotting has become considerably simpler than printing, since a complete page such as Fig.6 can be generated by one FORTRAN statement.

7. CONVERSION TO THE SC 4020

An illustration of the power of these plotting routines was noticed when the SC 4020 microfilm recorder was installed. Since all Culham graph-plotting still depended ultimately on the original subroutine PLOT, it was only necessary to rewrite this one subroutine for the SC 4020 in order to route all graphical output to the new medium. If on the other hand each programmer had written his own basic plotting instructions independently, an enormous amount of work would have been involved in changing to the new form of output. As a matter of fact, in common with some other computers STRETCH has the interesting property that if two or more subroutines are loaded which have the same name, the last one takes control and the others are ignored. Therefore to convert any existing program, (which might be on the PLT, with a structure quite unknown to the current user), it was strictly speaking unnecessary to alter the program at all, but simply to insert a single L-card representing

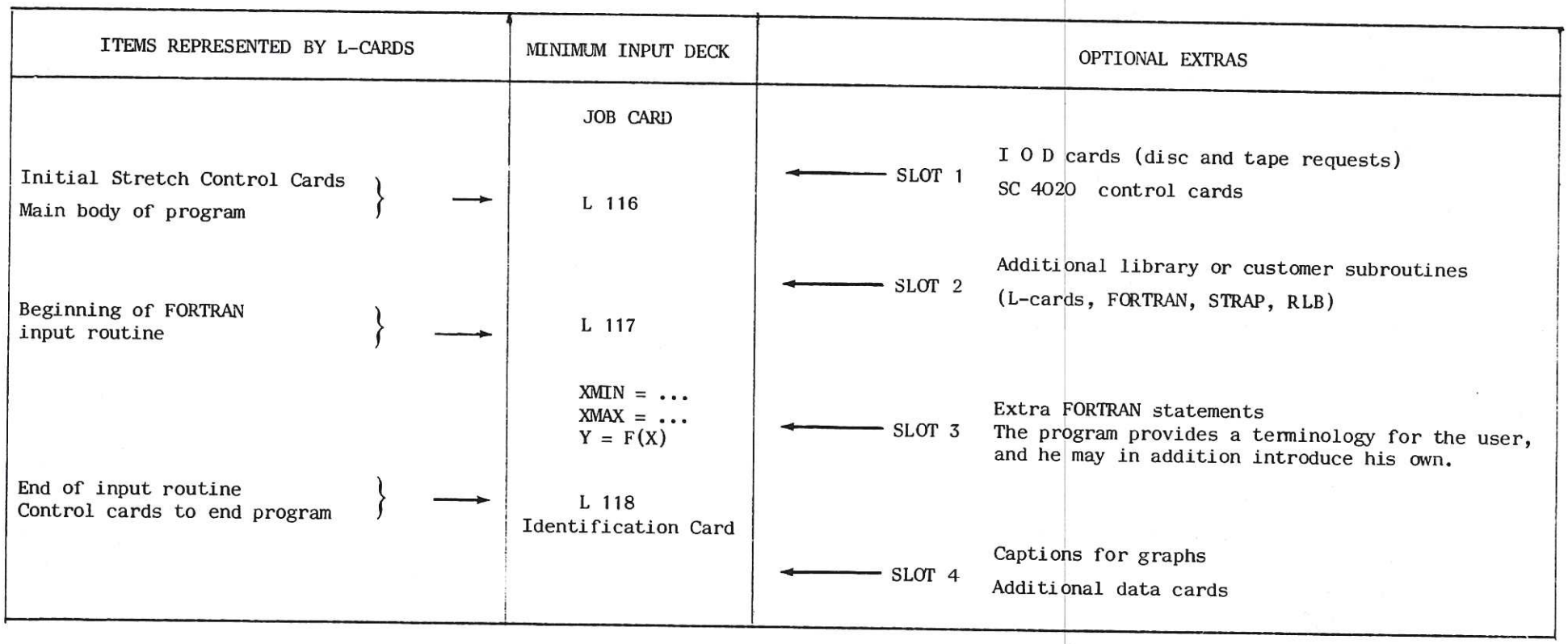


Fig. 9

(CLM-R45)

General structure of the input deck for Standard Program 1. Extra material may be added in any or all of the four slots, so extending the range of application of the program indefinitely

Two such programs have so far been written, designed to plot and tabulate arbitrary functions (or sets of functions) in one and two dimensions respectively. Fig.8 shows the card deck needed for the particular function $J_0(x)$, $(0 \leqslant x \leqslant 10)$. Fifteen cards are needed in this case, of which nine are prepunched and six are punched by the user*. All the standard part of the job deck is on the PLT, leaving only those features which are specific to the individual problem to be supplied explicitly. Fig.9 shows the general deck structure and Fig.10 is a simplified instruction sheet. The basic philosophy is to allow the computing system to exhibit a considerable amount of initiative, making its own choice for all but the essential data needed to specify the problem; if however the user wishes to make his own conscious decisions, he can insert FORTRAN statements in Slot 3 of Fig.9, and these will overwrite decisions made by the system, (e.g. method of plotting or tabulation, number of mesh intervals, accuracy etc.). By suitable FORTRAN statements inserted in this slot, almost any continuous mathematical function or set of functions may be defined. Altogether there are four slots in the deck shown in Fig.9, into which extra control cards, subroutines or L-cards, FORTRAN statements, and numerical data may respectively be inserted, so extending the range of application indefinitely. The program has in fact been used for several types of problem not foreseen by the original writer, such as the solution of differential equations and the evaluation of integrals.

At this stage, the simple task envisaged at the beginning of this paper has essentially been accomplished; it is indeed possible to use the computer system to plot the function $J_0(x)$ or any other continuous function, no matter how complicated its definition, simply by "typing" the request in mathematical form on a small number of cards, and "dialling" in the necessary programs and subroutines by selecting L-cards from a tray. For this type of problem, the computer has become almost as simple to use as an automatic telephone, and the facility has been in constant use since 1963 by members of the Laboratory with little or no knowledge of programming. Nevertheless because the STRETCH computer is not on site it was still necessary to wait at least three hours for the results, so that unless the function to be plotted was quite complicated it was not always worth using the machine.

Facilities at Culham are being steadily extended by providing more standard programs and subroutines, improving the documentation, reducing the turn-round time, and increasing the ease of access by the scientist. Ultimately we envisage very large libraries indeed, containing thousands of tested items stored on devices such as the magnetic card file, and

---

* By further optimization the number could be reduced to ten, (6 + 4).

TABULATE AND PLOT A FUNCTION

*CSP 001

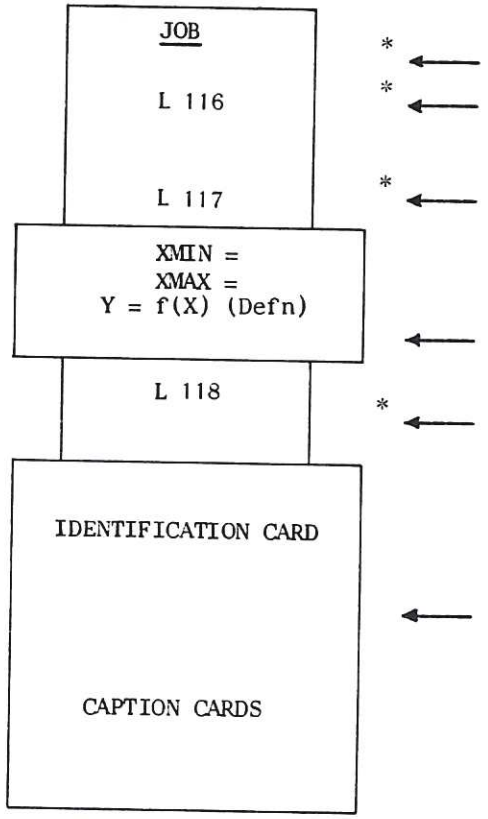
Ref. Nos. 116
117
118

Culham Standard Program No: 001

MINIMUM FORMAT

OPTIONAL

NECESSARY



Insert here any extra L-cards, binary or S2 routines needed to define the function

*Standard library cards, obtainable from reception desk. Do not punch.

Fortran (S2) cards, statement starting in column 7.

	Columns
Date	2/3,5/6,8/9
Initials	11-30
Description	31-70
Number of Caption Cards	72 (N = 0,1,2)

1 or 2 cards giving (i) horizontal and (ii) vertical captions. These must be punched in columns 2-72, running from first to second card if necessary. Each must be terminated with "£" (£) and neither may be more than 50 symbols long.

Fig. 10 (CLM-R45)
Simplified instruction sheet for using Standard Program 1, requiring no knowledge of programming. Progressively more detailed descriptions of the facilities provided by the program are available for those with more sophisticated requirements

maintained by teams of mathematicians and system programmers, since reliability and efficiency are crucial. But if this kind of scheme is to have a marked effect on the progress of scientific work, it is necessary for the scientist to be able to feed information to the machine via a key-board on his desk, and to have the results displayed as quickly as possible on a screen in front of him. This is the system pioneered at Space Technology Laboratories[5] and elsewhere in the U.S.A. Immediate access is even more necessary for efficient program development, since otherwise the mere omission of a comma can hold up work for hours.

10. THE KDF9 AND THE EGDON PROGRAMMING SYSTEM

To enable further improvements to be made in the system, it was necessary to have a computer configuration at Culham, and as a basis for such a configuration the Laboratory decided to install the English-Electric-Leo-Marconi KDF9 which is now in full operation. A large version was selected including a 32K core store and 4 million words of disc storage, together with 9 magnetic tape units and other peripherals. A disc-oriented programming system called EGDON, based on FORTRAN and punched cards, was worked out in collaboration between English Electric, Culham and Winfrith (who ordered an identical machine). It was implemented by the Company with some assistance from A.E.A. programmers. The EGDON system has now been working for some time, and details will be published in a separate report.

11. AUTOMATIC DOCUMENTATION

One of the main requirements was that the initial EGDON system should be capable of rapid and continuous evolution, in order to improve its efficiency from the point of view of the user in the light of experience, and to take advantage of new hardware and software developments such as on-line time-shared computing. As with any other expanding Laboratory service such as the electricity supply, it is essential for this purpose to have a thorough understanding of the current system, so that the detailed ramifications of any proposed changes can be quickly grasped. The concept of operational research had therefore to be extended to system programming. It is customary to write this type of software from flow-charts[10]; few programmers have yet become proficient at inserting enough comments into their actual code to make it easy for other people to follow (or even for themselves), and unlike FORTRAN and ALGOL the languages which are used for system work are in any case extremely difficult to read. When a large system is being changed frequently it is an expensive and almost superhuman task to maintain adequate documentation by normal office methods, but this should not prove difficult for a computer. The idea of automatic

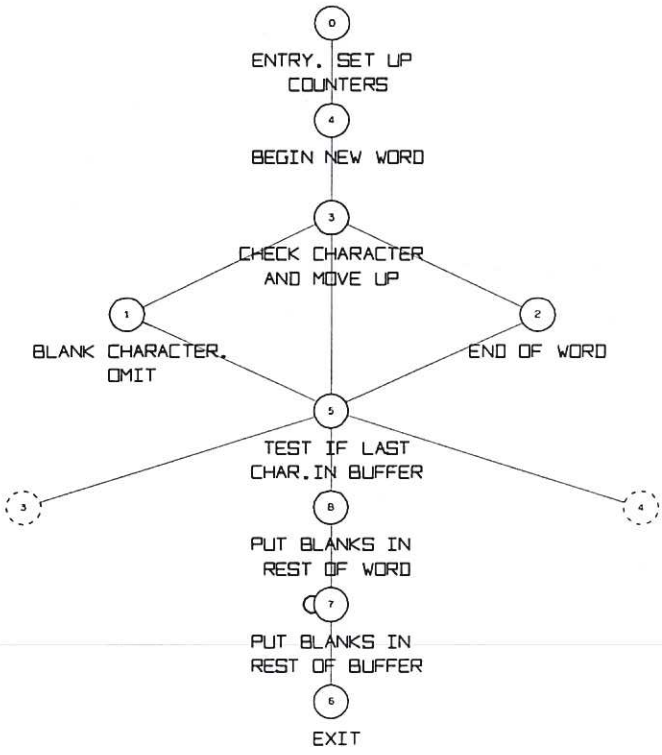


Fig. 11 (CLM-R45)
Annotated flowchart of one of the subroutines of the EGDON system, produced on the SC 4020 by the Hain flowcharter. Control runs downwards. A dashed circle indicates a label which has already occurred at a higher level

documentation was therefore formulated; system software is to be organised in such a way that flow-charts, annotation and indexes can be generated by the computer itself, solely from the original code and its inserted comments, and brought up to date automatically as soon as any alteration is made.

The term "meta-software" has been coined to denote programs that analyse other programs, and three examples have so far been developed at Culham. One of these is a flow-charter, written by K. Hain and G. Hain and further adapted by S.J. Roberts. Fig.11 shows an annotated flow-chart which has been produced by this program on the SC 4020, from the original KDF9 USER CODE of a short subroutine of the EGDON system, while Fig.12 shows a much larger chart, so far without

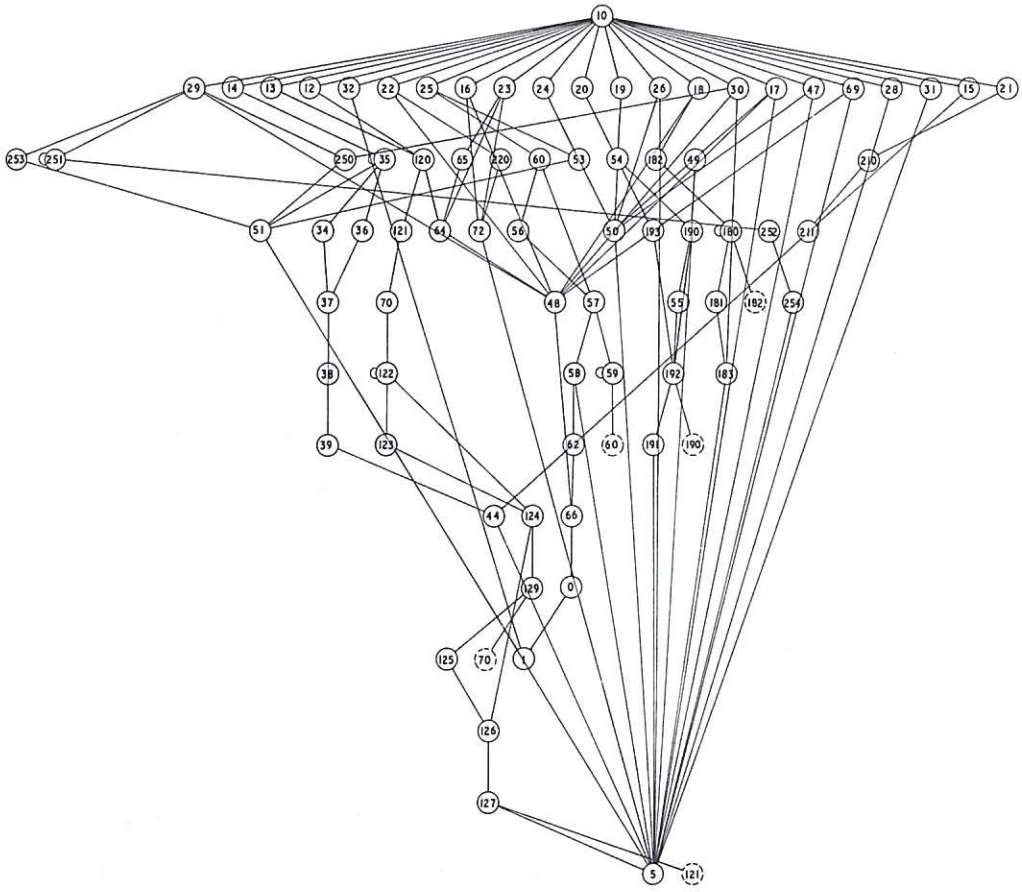


Fig. 12 (CLM-R45)
The main subroutine of the EGDON Job Organiser, from a flowchart produced on the Benson-Lehner plotter

annotation. The comments in Fig.11 have been picked up from a separate table, but in a fully-organised scheme they would be included in the code itself as it was being written. The second example is a documenter, started by the author and developed by D.L. Fisher, which processes a complete USER CODE program deck, commenting on all variables and jumps and compiling appropriate indexes. All the EGDON software is being analysed in this way. The need for such automatic assistance in reading large system programs may be seen from Table 3 which is one of the subroutines of the EGDON Job Organiser and represents possibly 0.2% of the entire system. Using automatic documentation, two substantial improvements (from the point of view of the user) have already been included in the EGDON system; these were made in a few hours without programming errors. Normally it is extremely difficult for customers to alter large system programs.

Finally, a flowcharter to make FORTRAN programs more readable has been developed by J. Staples and the author. Fig.13 shows one of the subroutines of the flowcharter itself, analysed in this way.

12. OPERATIONAL RESEARCH INTO SOFTWARE DEVELOPMENT

The real economic importance of system software is now being recognised, but little emphasis has yet been placed on finding out how to produce and maintain it more efficiently, in spite of a serious shortage of system programmers, and of the embarrassment caused to manufacturers and customers by late delivery. The following topics are suggested for operational research:

A. Design an optimum set of automatically-generated indexes, annotation and diagrams which will make system programs as easy as possible to understand, (analogous to maps, gazetteers, directories etc.).

B. Establish a Code of Practice for system programmers, containing rules for documentation which are adapted to automatic techniques.

C. Develop a proper language for system programming, which is easy to write and especially to read, and which is efficient in both machine time and instruction space.

D. Introduce a character set adequate to express this language.

E. Design on-line hardware which will make the work of the system programmer as simple as possible.

F. Develop techniques which exploit this hardware; for example by following the working of a program on-line, using charts and diagrams displayed on a screen.

G. Monitor the working of the system by inspection programs, in order to measure its efficiency, and to see how it can be further improved.

```
        SUBROUTINE NUMBER (KN,KS,KF,KC)

        ..... common-dimension list omitted in reproduction .....

        ****** READ NUMBER KN, STARTING IN COLUMN KS.  NO BLANKS ALLOWED WITHIN ******
        ****** OR TO RIGHT OF NUMBER ******
        ****** NEXT CHARACTER IS KC, IN COLUMN KF ******

        KN=0
           +
        ****** READ BLANKS UNTIL FIRST SYMBOL REACHED ******
  $-----DO 1 K = KS, 72
  L        +
+·L·····IF (NCARD (K) -MBL)2,1,2
V L        +
V +...1 CONTINUE
V          +--------$
V                ****** END OF CARD ******
V       $--------51 KF = 73
V       A           KC = MBL
V       A           RETURN ----------------------------------------------------------------------------------------------------RETURN
V       A
V       A                   ****** SYMBOL FOUND ******
$------A-----------------2 K1=K
        A                      +
        A                   ****** ACCUMULATE NUMBER ******
        A         $---------- DO 7 K=K1,72
        A         L           KD+NCARD(K)
        A         L           ****** IS THE CHARACTER A NUMBER ******
        A         L +.+..... IF(KD-M1)5,4,3
        A         L V V         +
        A         L V +....3 IF(KD-M1-8)4,4,5
        A         L V V         +
        A         L V V       ****** YES.  LYING IN RANGE 1 - 9 ******
        A         L $-V-----4 KN = 10*KN +KD-M1+1
        A         L   V         +
        A         L +.V..... GO TO 7
        A         L V V
        A         L V V               ****** IS THE CHARACTER ZERO ******
        A      +·L·V·$==========5 IF(KD-MZERO)8,6,8
        A      V L V                    +
        A      V L V                    ****** YES ******
        A      V L V                 6 KN=10*KN
        A      V L V              $.......+
        A      V +·$======7 CONTINUE
        A      V                    +
        A      V             ****** END OF CARD REACHED ******
        +.....V..........  GO TO 51
               V                    ****** LAST CHARACTER REACHED ******
               $----------------- 8  KC=KD
                                    KF=K
                                    RETURN ---------------------------------------------------------------------RETURN
```

Fig. 13 (CLM-R 45)

One of the subroutines of the Fortran Flowcharter, automatically flowcharted by the program itself. (Copied from the computer printout, with minor corrections.) Symbols A, V denote upward and downward paths, L an upward return in a DO-loop, ... and ----- transfers to left and right respectively, ===== both ways. Flow paths are entered at + and left at $. The symbol + also denotes a continuation in the same vertical column. Any comments included in the text are emphasised by asterisks. A well-written program will include a comment for nearly every statement number, and these should be inserted as the code is being written down. Routines of up to 500 cards may be handled.

A large system program is a valuable mathematical or logical document, and as much care should be devoted to improving its readability as one would give to an important journal article or text-book. If this is not done, major weaknesses are likely to go undetected or be difficult to remove, and systems are now developing so rapidly that anything which is unreadable and therefore incapable of continuous evolution is likely to be abandoned quite soon in favour of an entirely new version, at a considerable waste of scarce software effort.

Good documentation is relatively easy to achieve by automatic methods, once the problem has been recognised. Starting from the actual code, it should not prove difficult to produce a complete 'hand-book' on the line printer, or on a device such as the SC 4020 in hardcopy or microfilm form, including annotated code, diagrams, page headings, indexes, footnotes and any other features that may be found useful. It is worth mentioning the complementary problem of the language in which system programs are written - this deserves a considerable amount of thought, because evidently a good language or notation can make an enormous difference in mathematics; consider for example the invention of the zero in arithmetic, or the use of Arabic rather than Roman numerals. At present, the development

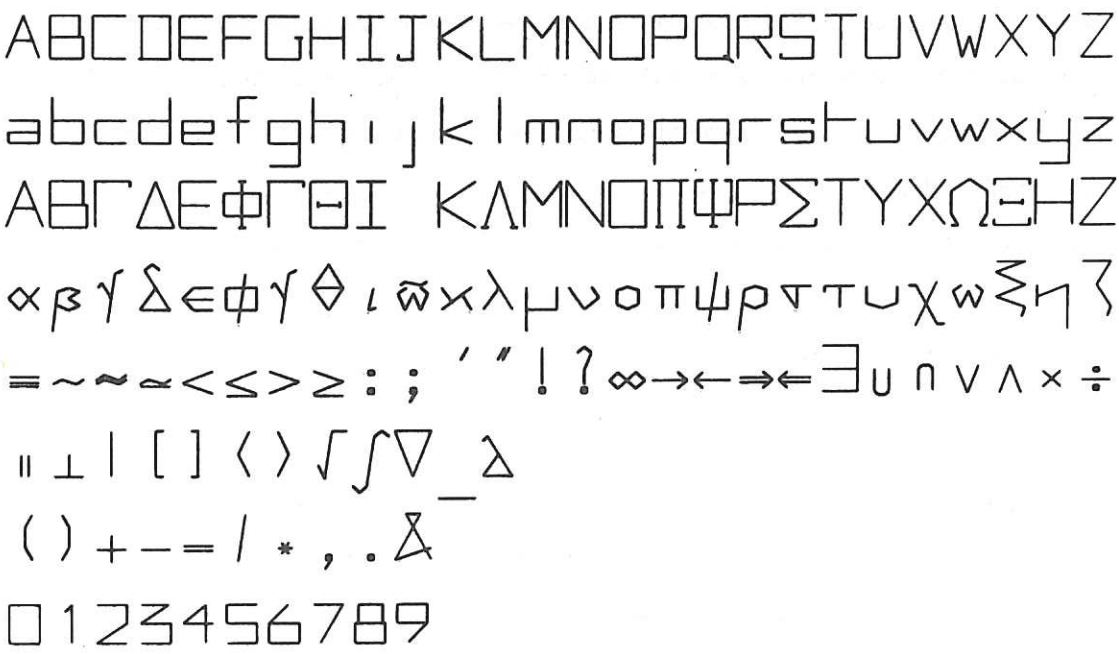


Fig. 14 (CLM-R45)
Programmed character set for visual display device

of programming languages is gravely hampered by a quite incidental difficulty; the inadequacy of the character set available on card punches and line printers. For a fraction of the cost of the current software effort it should be possible to introduce an extended set, comparable to that available on a mathematical double-bank typewriter. On the other hand, this difficulty may well be removed by the use of on-line key-boards and display screens in system programming, since this allows quite arbitrary symbols to be defined. Permanent listings can be produced on either hardcopy or microfilm. A subroutine which generates most of the symbols used in mathematics has been written by A.N. Dixit and the author and an example of SC 4020 output produced in this way is shown in Fig.14.

13. USER CODE AND SYSTEM CODE

As a start on the language problem, some work has been done in the direction of extending KDF9 USER CODE to form a more convenient system language, while accepting the restrictions imposed by the standard punched-card character set, and keeping the necessary translator as simple as possible. USER CODE is a very suitable starting point for this purpose, since apart from its Reverse Polish notation[8,11], it employs standard symbols or names for mathematical and logical operations, (+, -, /, NOT, AND etc.). The extended language, which is called SYSTEM CODE, is equally efficient in terms of machine speed and instruction space, but much more powerful and readable, since it includes mnemonic names, algebraic and logical formulae, argument lists, function subroutines, and macros, while descent into USER CODE is possible at any point. The programmer is able to define new compound instructions and macros as he goes along, and therefore virtually to develop his own private language, just like the author of any other mathematical text. All the declarations necessary for this purpose can then be incorporated in the Automatic Library, and so brought into play by a single L-card.

According to present proposals, a simple Translator I would be written in FORTRAN, to convert SYSTEM CODE into USER CODE. A new Translator II would then be written in SYSTEM CODE, converted by means of Translator I, assembled, and incorporated into the EGDON system. Parts of the EGDON system would then be translated literally into SYSTEM CODE by hand, in order to gain experience and to develop a suitable set of macros, and to make the system easier to understand. Further extensions to the system would subsequently be made in the new language.

14. CONCLUSION

In the past, system software has mainly been concerned with achieving an efficient utilisation of the machine and simplifying the work of the operators. Except for the introduction of higher level languages such as AUTOCODES, ALGOL and FORTRAN, it has done little for the user. These earlier problems have largely been solved and there is now a rich field for operational research, leading to automatic techniques which will make the computer system an ever more effective tool for the user, and increase the rate at which software itself can be developed. Apart from its intrinsic interest and application to scientific research, this type of work is likely to have considerable economic importance. Unlike other forms of automation, all the necessary hardware exists and forms an integral part of the computer system itself, so that progress should be rapid.

15. REFERENCES

1. McCARTHY, J. (1962) "Time-Sharing Computer Systems", in "Management and the Computer of the Future", M. Greenberger, ed., Cambridge, Mass. The M.I.T. Press, pp.221-236.

2. CORBATO, F.J. et al. (1963) "The Compatible Time-Sharing System - A Programmer's Guide", M.I.T. Press, Cambridge, Mass.

3. DENNIS, J.B. (1964) Comm. ACM 7, 521.

4. FANO, R.M. (1965) I.E.E.E. Spectrum (January issue, p.56).

5. CULLER, C.J. and FRIED, B.D. (1964) STL Report 8587-6002-RV000.

6. FIELD, E.C. and FRIED, B.D. (1964) Phys. Fluids, 7, 1937.

7. RIGG, F.A. (1964) Computer Journal, 7, 169.

8. WEGNER, P. (1963) "Introduction to System Programming", Academic Press, London and New York.

9. LARKIN, F.M. (1964) Computer Journal, 7, 212.

10. GOLDSTINE, H.H. and VON NEUMANN, J. (1947) "Planning and Coding of Problems for an Electronic, Computing Instrument", reprinted in John Von Neumann "Collected Works", Vol.V. Pergamon Press, Oxford, 1963.

11. DIJKSTRA, E.W. (1961) Algol Bull. Suppl. 10.

TABLE 1

RECENT DEVELOPMENTS IN COMPUTER HARDWARE

TYPE	EXAMPLES	NOTES
Core Storage		Up to 1 million words Cost $\leq$ £1/word Minimum transfer time 1/40 μsec
Random Access Backing Storage	Magnetic card file Data cell	60 million characters Cost ~ £70,000 Access time 1/3 - 1/2 sec
Visual Output	Graph plotter Microfilm recorder Hardcopy recorder On-line CRT	Arbitrary combination of printed and diagrammatic material, including ciné film
Visual Input	Light pen Microfilm reader	
On-line Computing	Typewriter Light Pen Cathode ray tube Programmed Key-board	Minimum turn-round time 1-5 seconds Screens can display 8000 characters, and graphs
Telecommunications	Computer - Computer Subscriber - Computer	In principle, any two units can be connected via the public international telephone network, and automatic dialling can be programmed
Large Computers	CDC 6800 IBM 360/95	Speed possibly 50 x IBM 7090
Small Computers	PDP 8	Minimum cost ~ £7000.

TABLE 2

CHARACTERISTICS LISTED IN CHART

STATUS	O-operational system, number in parenthesis denotes the approximate date that the system went on the air. D-system under development with anticipated date that operations will begin.
TYPE	G-general purpose, S-special purpose.
COMPUTER	manufacturer's name and number of central computers in system.
C/M/U/N	denotes whether commercial, military, university or non-profit organization operates system. PR-denotes system for private or internal use only, PU-system available for general public use, SP-semi-public use permitted.
LANGUAGES	basic languages available on system at present.
TERMINALS	type of terminal equipment available, number of such terminals in parenthesis. Code: TT followed by number denotes TELETYPE terminals and model number, TY-typewriter, TLX-Telex console, CRT-cathode ray tube display, BR-Bunker Ramo series 200 display consoles, IBM 1050-keyboard consoles.
MAIN STORAGE	first number denotes total core storage on system, second number in parenthesis, if given, denotes maximum core storage available to an individual user.
SECONDARY STORAGE	DR-magnetic drum, DK-disk file, MT-magnetic tape (K = 1000, M = 1,000,000).
NO. OF USERS	maximum number of users who can operate simultaneously at any given time.

TIME-SHARING SYSTEM SCORECARD

Prepared by COMPUTER RESEARCH CORPORATION

ORGANIZATION	STATUS	TYPE	COMPUTER(S)	C/M/U/N	LANGUAGE(S)	TERMINALS	MAIN STORAGE	SECONDARY STORAGE	NO. OF USERS	REMARKS
Adams Associates — Keydata System Cambridge, Mass.	O (5/65)	G	PDP-6	C-PU	Fortran KOP-III	TT-28 (16)	48K (32K)	DR (1M Wds.) DK (11M Wds.) MT (2 Units)	16 [1]	For on-line invoice preparation and inventory control, other accounting uses under development
Aviation Supply Office[2] Philadelphia, Pennsylvania	O (10/62)	S	IBM-1410	M-PR		IBM-1014 (12)		DK (2 Units)	2	Inventory control system
Bell Telephone Laboratories[3] Murray Hill, New Jersey	D (2/66)	G	GE-636 [4]	C-PR		Information not available				
Bolt Beranek and Newman Inc.[5] Cambridge, Mass.	O (6/64)	G	PDP-1D [6]	C-SP	MIDAS TOLL-1 [7]	TT-33 (48)	24K (4K)	DR (128K Wds.) DR (25M Wds.) MT (2 Units)	32	Medical Information and communications system for hospitals
Carnegie Institute of Technology Pittsburgh, Penn.	O (3/65)	S	2 G-20	U-PR	ALGOL	TT-33 (12)		DR	12	
Dartmouth College[8] Hanover, N. H.	O (9/64)	G	GE 235 DATANET-30	U-PR	BASIC ALGOL	TT-35 (22)	(6K)	DK MT	8	Educational time-sharing system
IBM QUICKTRAN Service New York, New York	O (5/65)	S	IBM-7040 7044	C-PU	QUICKTRAN [9]	IBM 1050 (40)	32K	DK MT	40	On-line scientific computation service
MIT Computation Center Cambridge, Mass.	O (9/63)	G	IBM-7094	U-SP	Same as Project MAC Phase one		64K (32K)	DK DM MT	Same as Project MAC Phase one	
MIT Dept. of Electrical Eng. Cambridge, Mass.	O (9/63)	G	PDP-1	U-PR	MIDAS	TY (3)	4K	DR	3	Experimental time-sharing system for student use
Naval Command System[10] Support Activity	O (12/64)	S	2 CDC-1604 2 CDC-160A	M-PR		TT-33 (8)	32K	DK (2 units)	8	For tracking, control and scheduling naval vessels
Project MAC — MIT (Phase One) Cambridge, Mass.	O (9/63)[11]	G	IBM-7094	U-SP	ALGOL[12] FORTRAN MAD LISP	TT-35 (54) IBM-1050 (56) TLX (1)	64K (32K)	DR (36M Wds.) DM MT	30	Project MAC is an MIT research program sponsored by the Advanced Research Projects Agency, D.O.D., under the Office of Naval Research
Project MAC — MIT (Phase Two) Cambridge, Mass.	D (2/66)	G	GE-636 [4]	U-SP		500 [13]	128K	DK DM	150 [13]	
RAND Corporation Santa Monica, California	O (2/64)	S	Johnniac	N-PR	JOSS	TY (8)	4K Wds.	DR (12K Wds.)	8	
Space Technology Laboratory El Segundo, California (Culler-Fried System)	O (1/65)	S	Bunker-Ramo 340	C-SP	MATHE-MATICAL ANALYSIS	4 Consoles [14]	8K	DR (48K Wds.) MT	4	Highly flexible system for on-line manipulation, specification and execution of mathematical operations with graphical display of results
Stanford University Stanford, California	O (6/64)	G	IBM-7090 [15] PDP-1	U-SP	MACRO [16] LISP FORTRAN	PHILCO (12) TT (8)	20K	DK DR	20	
System Development Corp. Santa Monica, California	O (6/64)	G	AN/FSQ-32 [15] PDP-1	N-SP	TINT IPL-TS JOVIAL	TT-28 (8) TT-33 (16) TY (3) CRT (6)	80K (48K)	DR (400K Wds.) DR (4M Wds.) MT (16 Units)	30	Oriented to command and control experimentation
U.C.L.A. Western Data Processing Center Los Angeles, California	O (11/64)	S	IBM-7740 [17] IBM-7040/ 7094	U-SP		IBM-1050 (12)	32K	DK DM	12	Jointly financed by UCLA and IBM, system services UCLA and 88 other California schools
University of California Berkeley, California	D ( )	G	SDS-930	U-PR	FORTRAN	TT-33 (6) CRT	32K	DR	6	
University of California Santa Barbara, California (Culler-Fried System)	O (3/65)	S	RW 400 AN/FSQ-27	U-PR	MATHE-MATICAL ANALYSIS	16 Consoles [14, 18]	6K	DR (80K Wds.) DR (500K Wds.)	16	Highly flexible system for on-line manipulation, specification and execution of mathematical operations with graphical display of results
University of Pennsylvania Philadelphia, Penn.	D (6/65)	G	IBM-7040 PDP-5	U-SP	MULTI-LANG MAP ALGOL	TT-35 (4) BR (2)	32K (24K)	DK	6	

NOTES

1. System to be expanded to 48 users shortly.
2. Developed under contract with the Moore School of Electrical Engineering, University of Pennsylvania.
3. Development in cooperation with Project MAC, Massachusetts Institute of Technology.
4. Multiple Processor Time-sharing system.
5. Developed with the Massachusetts General Hospital under contract from the National Institutes of Health.
6. Based upon an earlier 5 station PDP-1 system operational 9/63.
7. Version of the RAND JOSS language.
8. Developed with the cooperation of the General Electric Co.
9. On-line version of FORTRAN.
10. Developed under contract with Computer Command Control Corp.
11. Initially time-shared in 1962 at the M.I.T. Computation Center.
12. Other languages include FAP, SLIP, COGO, SNOBOL, STRESS and OPL-1.
13. This system is eventually expected to handle 500 terminals and 150 simultaneous users some time after initial operation begins.
14. Each console consists of two keyboards and a storage tube display. A camera and plotter are shared among the consoles.
15. Example of main computer with satellite computer for communication with consoles and scheduling.
16. Other languages include FAP, GOGOL and BALGOL.
17. System currently utilizes five computers in addition to central 7740.
18. Other terminal equipment to be installed include a RAND tablet and a Grafacon.

Spring 1965. Reprinted by permission of Computer Research Corporation, Belmont, Mass. U.S.A., from whom subsequent editions are available.

TABLE 3

SUBROUTINE 702 OF THE EGDON JOB ORGANISER IN USER CODE

```
                                                                                %701999;
P702,  %HANDLE JOB CARD;,  SET6, =RC15,                                         %702000;
*,1, V0P700M15, =V82P700M150, J1C15NZS, SET114, OUT, ERASE, SHL-16,              %702100;
     SETB63607, SHL30, OR, =V80P700, V63P700, SET103, OUT,                      %702150;
     V21P797, DUP, =RM14, =RM15, SET4, =RM7, JS8, JS8, JS6P704, J6,             %702200;
     =V6P700, JSP704, ERASE, V6P700,                                            %702210;
7,   JS6P704, J9, =V7P700, JSP704, ERASE, V7P700,                               %702220;
10,  JS13, JSP710,                                                              %702230;
12,  DUP, ZERO, SHLD12, J11=Z, ERASE,                                           %702240;
14,  SETB1400, CAB, OR, REV, =M0M150, =M0M150,                                  %702245;
     JS42P700, SET60, XD, CONT, SHL24, SET114, OUT, DUP, JSP710,                %702250;
     =M0M150, =M0M150, ZERO, SHLD30, =V6P700, SHL-12, SHLD12, =V7P700,          %702300;
     SETB202, OR, =V8P700, NC15, C15TOQ14, Q14, SET115, OUT,                    %702350;
     V0P700, SHL6, SET3, OR, SHC-6, =V0P700, V34P700, NEG, NOT, DUP,            %702400;
     JS2, JS33P700, JS3, JS4, JS3, JS2, EXIT1,                                  %702450;
8,   JSP704, ERASE,                                                             %702500;
13,  JSP710, SETB1400, SHC-12, OR, =M0M150, =M0M150, EXIT1,                     %702550;
9,   ERASE, JS13, ZERO, ZERO, J14,                                              %702600;
11,  ERASE, SHLD12, J12,                                                        %702700;
6,   ERASE, ZERO, REV, J7,                                                      %702800;
2,   SET6, SHL32, SET240, OR, J4,                                              %702910;
3,   V0P798,                                                                   %702920;
4,   SET123, J43P700,                                                          %702930;
```

Subroutine from EGDON Job Organiser

Note

This example shows how difficult it can be to read a program written in symbolic language with any degree of comprehension of the overall meaning - the individual instructions are clear enough. A detailed analysis of the readability problem will be given in a subsequent report, comparing the same subroutine in various forms, e.g. USER CODE with and without documentation and SYSTEM CODE.

Available from

HER MAJESTY'S STATIONERY OFFICE

49 High Holborn, London, W.C.1
423 Oxford Street, London W.1
13a Castle Street, Edinburgh 2
109 St. Mary Street, Cardiff
Brazennose Street, Manchester 2
50 Fairfax Street, Bristol 1
35 Smallbrook, Ringway, Birmingham 5
80 Chichester Street, Belfast

or through any bookseller.

Printed in England

S. O. Code No. 91-3-18-43