UNIVERSITY OF QUEENSLAND
COMPUTER CENTRE

COMPUTER CENTRE BULLETIN

Vol. 4, No. ~~6.~~ 7
5 July 1971

Editor:
Mrs Sarah Barry

COMPUTER CENTRE COURSES FOR 1971

The attached schedule details the courses which will be given by the Computer Centre in the last six months of this year. Enrolments should be made on the Computer Centre Course Nomination Form and be forwarded to the Computer Centre before the closing date for that course.

If nominations for any course are insufficient, the course may be cancelled. A brief description of each course is given below.

1. INTRODUCTORY FORTRAN PROGRAMMING

Two courses in the FORTRAN programming language will be given. These courses are introductory courses and assume that the student has no prior knowledge of computers or programming. Students will be taught elementary FORTRAN programming and will run a number of exercises on the PDP-10 computer.

2. DDT - DYNAMIC DEBUGGING TECHNIQUES

DDT is a powerful means of debugging a program for use in interactive mode. The one day seminar will introduce remote terminal users to the concepts and basic use of DDT, and provide a limited amount of practical experience.

3. STATISTICAL PACKAGES

A course will be given on Statistical Computation by Mr I. Oliver. The course will be based on the BMD series of packages, many of which will be made available on the PDP-10. Attendees will be expected to be familiar with elementary statistical concepts including probability, significance testing, means, variances, correlation and chi-square. It would be an advantage if some prior contact was had with multiple linear regression, factor analysis and analysis of variance.

4. INTRODUCTORY BASIC PROGRAMMING

The BASIC programming language is a simple, FORTRAN-like language, designed specifically for interactive use via a remote terminal. This course introduces users with some experience of FORTRAN programming to the BASIC language and use of remote terminals.

5. INTRODUCTORY MACRO PROGRAMMING

MACRO is the assembler language for the PDP-10. This course will describe some aspects of the internal operation of the PDP-10 machine, introduce assembly language programming, outline the MACRO instructions available and consider the writing of macros and subroutines for use with FORTRAN mainline programs. As the course is introductory, it will not treat Input/Output operations in MACRO, or any of the more advanced features of the language. Knowledge of FORTRAN for the PDP-10 is a prerequisite for this course.

Course	Date	Time	Location	Fee* (incl. machine time)			Enrolment Closing Date	Prerequisite Knowledge
				UNI	GOVT	EXT		
Introductory FORTRAN Programming (July course)	12-29 July (Monday & Thursday evenings only)	7 - 10 pm (Monday & Thursday)	B18 Main Engin. Building	$20	$40	$50	5 July	None
Introductory FORTRAN Programming (October course)	25-29 October	2 - 6 pm	B18 Main Engin. Building	$20	$40	$50	18 October	None
Use of Dynamic Debugging Technique (DDT)	16 August	9am - 5pm	B18 Main Engin. Building	$ 8	$16	$20	9 August	Use of FORTRAN and remote terminal on PDP-10
Statistical Packages (including BMD packages)	23-27 August	9am - 1pm daily	B18 Main Engin. Building	$20	$40	$50	16 August	Elementary Statistical Concepts
Introductory BASIC Programming	9&16 September (2 Thursday evenings)	7 - 10 pm	B18 Main Engin. Building	$ 8	$16	$20	2 September	Some FORTRAN Experience
Introductory MACRO Programming (One course covering 6 mornings)	25-27 October 1- 3 November	9am - 1pm	B18 Main Engin. Building	$24	$48	$60	18 October	PDP-10 FORTRAN

* In all cases, the course fee includes provision of machine time for exercises. The three categories in the scale of fees are

UNI: University Departments
GOVT: Government Organizations
EXT: Non Government Organizations

AUSTRALIAN COMPUTER SOCIETY (ACS) OVERSEAS VISITORS PROGRAMME

The second distinguished visitor to be brought to Australia under the ACS Overseas Visitors Programme will be Professor Bernard A. Galler, Professor of Computer and Communication Sciences and Mathematics and Associate Director of the Computer Centre at the University of Michigan. Professor Galler has a wide background in computing as shown by the fact that he was President of the Association for Computing Machinery (ACM), 1968-1970, and a member, Board of Governors, American Federation of Information Processing Societies (AFIPS), 1968-1970. His publications include 'The Language of Computers', 'A View of Programming Languages', and several articles on automatic programming and linear programming.

The Australian Post Office and IBM Australia have very generously agreed to make the arrangements for Professor Galler's Michigan Terminal System (MTS) demonstrations in Brisbane. The demonstration will involve use of IBM's 360/67 in Canberra and A.P.O. lines and modems.

Professor Galler will deliver a public lecture and conduct a one-day seminar this month.

LECTURE - Wednesday 28 July 8.00 p.m. The lecture will be entitled 'A view of programming languages' and will include a general review of presently available languages and possible future developments.

Hawken Auditorium, The Institute of Engineers, 447 Upper Edward Street.

SEMINAR - Thursday 29 July 9.30 a.m. to 4.30 p.m.

Australian Institute of Management, Management House, Boundary and Rosa Streets, SPRING HILL.

The general subject of this seminar will be 'A demonstration of the Michigan Terminal System'. Topics will include a discussion of some desirable properties of a general purpose timesharing system and of the operating characteristics of such a system. A feature of the seminar will be the demonstration of the system using IBM's 360/67 in Canberra.

There will be a fee of $1 for the lecture which includes supper and a nominal contribution to assist the Programme's expenses. For the seminar, the fee will be $11 for ACS members and $16 for others, including a buffet lunch and drinks at the conclusion of the day. Persons wishing to attend the seminar are requested to complete an application form available from the Hon. Secretary, ACS (Qld Branch), G.P.O. Box 1484, BRISBANE, 4001.

LINE PRINTER AVAILABLE TO REMOTE TERMINAL USERS

Work on the first version of the line printer symbiont (spooling) system is now complete. This system has been implemented and will be available to remote terminal users from Tuesday 15 June. Users can transmit ASCII files to the printer symbiont for later printing by means of a new command LIST.

LIST command

The general form of the command is:

```
LIST (F4) filename-1, ..., filename-n
```

The arguments, filename-1, ..., filename-n, give the names of the ASCII files to be printed.

The option F4 is used in the command to print ASCII files which have been output by a user's FORTRAN IV program and which contain the standard line printer carriage control characters used in FORTRAN (ref. FORTRAN IV Manual MNT-5, page 6-17). The option can be associated with the command, to apply to all files named in the argument list, or can be associated with individual files in the argument list.

examples:

(i) To list two source files named PROG/F4 and TEST/MAC on the line printer

```
.LIST PROG/F4, TEST/MAC<cr>
-LISTING THESE FILES-

PROG/F4
TEST/MAC

EXIT
↑C

.
```

(ii) To list an ASCII file named DOUT which has been produced by a FORTRAN program and contains FORTRAN printer carriage control characters.

```
.LIST (F4) DOUT<cr>
-LISTING THESE FILES-

DOUT

EXIT
↑C

.
```

(iii) The three files contained in examples (i) and (ii) above could be listed with the one command as follows:

```
.LIST PROG/F4, TEST/MAC, DOUT (F4)<cr>
-LISTING THESE FILES-

PROG/F4
TEST/MAC
DOUT

EXIT
↑C

.
```

Error Messages on Printed Output

The following error messages could appear on the printed output.

(a) *** PRINTING TERMINATED BY OPERATOR ***

There are two possible causes of this:

(i) The job was slewing an unreasonably large number of forms without any printing. The printout will not be repeated.

(ii) The line printer damaged the output while printing. This file will be repeated.

(b) *** ERROR READING DATA FILE ***

This results from an error on the disk file being printed. The operator is informed of this error and action will be taken to attempt to recover the file for later printing.

Administrative Arrangements

Output from the line printer is identified by name bands on the first and last pages (in the same way as batch output) and can be collected from the PDP-10 output shelves. It is anticipated that output will be available for collection within a few hours of the user listing the file. All printing tasks will be automatically charged at the published rates.

With the release of this facility, the temporary file listing service for remote terminal users will be discontinued.

FORTRAN MANUAL

There is an error in the FORTRAN manual, MNT-5.

The first example on page 6-5 should read:

```
            READ (KRD,11) ((MASS (K,L), K=1,3),L=1, 5)
```

instead of

```
            READ (KRD,11) ((MASS (K,L), K=1,3) L=1, 5)
```

UTILITY PROGRAMS MANUAL

The Utility Programs manual, MNT-12, is now available at the University Bookshop. At present it contains details of Absolute Overlays and the Plotter subroutines and these chapters are available for $1.20.

USE OF PLOTTER AND OVERLAYS

Users are advised that both the overlay and plotter subroutines cannot be used in the same program.

This is a temporary restriction which will be removed shortly.

FORTRAN ERRORS

(a) If a unit number in a FORTRAN READ or WRITE statement is subscripted with a variable which is used also in the statement as a subscript, results become unpredictable. During the code produced by the statement the register containing the value of the variable is destroyed and hence the subscripted quantities will contain unpredictable numbers.

examples:

```
(i)   READ (K)   N,(IX(K), IN(J), IB(K), J = 2,N)
(ii)  READ (IUNIT(L))  N,(IN(J), IB(K), J = 2,N)
```

Both the above examples are correct. The following example is invalid.

```
(iii) READ (IUNIT(K)) N,(IN(J), IB(K), J= 2,N)
```

(b) The FORTRAN compiler recognizes tabs in FORMAT statements and inserts them as tab characters (not the equivalent number of spaces) into format strings. The FORTRAN operating system, however, converts these tab characters to single blanks on output, rather than 8 blanks per tab as might be expected.

<u>example</u>:

```
FORMAT ('A<tab   >A<tab   >A')
```

will produce an output

```
A A A
```

(c) The FORTRAN compiler does not keep track of jumps that might take it outside the range of the current DO loop. In cases of the extended range of a DO loop containing another DO loop, as in the following example, certain registers are not restored from memory after returning for the extension of the DO range.

<u>example</u>:

```
      DO 5Ø  I = 1,1Ø
      IF (I-5) 5Ø, 7, 5Ø
5Ø    CONTINUE
      STOP
 7    DO 88 K = 1,1Ø
      KK = K*1Ø
88    CONTINUE
      GO TO 5Ø
      END
```

(d) The compiler evolves coding that is only good for floated integers of 27 (or less) bits in the implied long-precision float operations:

```
A = N
```

For integers greater than 27 bits the compiler zeros the second word of the double precision scalar

```
DOUBLE PRECISION B
B = N
```

(e) The error message

```
?FILENAME FORTR NOT ON DEVICE DISK
```

is caused when the FORTRAN operating system cannot find a file with the name requested. Either the file does not exist at all, or more particularly, users are using file names with 6 characters when the operating system only recognizes the first five characters of the name.

(f) In some cases mixed mode arithmetic expressions do not compile correctly. In particular, the following expression,

```
D=S*D/(2*I-1)
```

where D and S are double precision and I is an integer, is known not to create proper code.

It is worth noting that the U.S.A. Standard FORTRAN specifications do not permit mixed mode expressions, and while several compilers do permit them, it is neither desirable nor necessary to make use of this facility.

(g) Users are reminded that use of large arguments for the sin and cosine functions may return values having appreciable errors in the sixth and succeeding figures.

(h) The compiler will not produce an I-5 error message (NAME ALREADY USED AS NAMELIST NAME) for the following coding:

```
NAMELIST/FRED/F,G

FRED = 3.0
```

Instead it will accept the redefinition of FRED as a scalar variable. Any attempt to use FRED as a namelist name will result in an I-4 error message (NOT A VARIABLE FORMAT ARRAY).

(i) The compiler treats digits that fall in the continuation field as a continuation indicator, regardless of the content preceding the continuation field.

example:

In the following coding, the statement READ 1Ø was ignored since the 1 was taken to be a continuation indicator. The number N was taken to be 160, instead of simply 16. The compiler failed to record any errors.

```
      N=16
READ 1Ø
 1Ø   FORMAT (  )
```

COBOL ERRORS

(a) A COMPUTE statement with 5 nested pairs of brackets causes the COBOL compiler to get a Push Down List overflow.

(b) The special data name TALLY is not always recognized as a defined field. Statements such as

```
ADD n TO TALLY

SET TALLY UP BY n

MOVE n TO TALLY
```

will not compile. Use of TALLY should be avoided for the present.

(c) Negative comparisons do not work correctly. For example the following coding

```
.
.
.
77 CON5 PIC S99 VALUE -31.
.
.
.
COND3.
  IF CON5 GREATER THAN -4Ø GO TO XYF
          ELSE GO TO NOK2.
.
.
.
```

will result in the program branching to NOK2 although -31 is greater than -40.

(d) In standard format, if the first character of a line is a tab, then this character is assumed to be at the 7th position, and the next character is assumed to be at the B-margin.

Items which should commence at the A-margin and are entered as above with a tab preceding them will not be recognized.

<u>example</u>:

```
<tab      >IDENTIFICATION DIVISION.
```

will cause the error message

```
IDENTIFICATION EXPECTED
```

BASIC ERRORS

(a) If an unquoted string containing an apostrophe is typed in response to an INPUT statement, only the part of the string up to the apostrophe is accepted.

example:

```
        ?"TEST APOSTROPHE I'M TESTING"
  will produce the complete message, but
        ?TEST APOSTROPHE I'M TESTING
  will produce only
        TEST APOSTROPHE I
```

(b) If an altmode is typed anywhere in an INPUT string the entire string is disregarded and the message DELETED appears.

example:

```
        ?TEST ALTMODE<altmode>DELETED
```

TELETYPE SUPPLIES

The following Teletype stationery items are available from the stationery store.

Code No	Description	Unit of Issue
17590	Paper tape 1" 8 Channel	1
17604	Paper rolls Teletype $8\frac{7}{16} \times 4\frac{1}{2}$	1
17612	Ribbons model 33/35 TTY	1
17620	Paper Fanfold Continuous $8\frac{1}{2} \times 11$	box (3000 sheets)

MOO

MOO is an intelligent guessing game. MOO chooses a random 4 digit number with all digits different and keeps the number to itself. The player has to find the number in the minimum number of guesses. MOO will help the player along by telling him how good his guess was after each guess.

After a guess, MOO responds with the number of 'bulls' and 'cows' scored. A *bull* is a direct hit; a correct digit in the correct position in the number. A *cow* is an indirect hit; a correct digit but in the wrong position in the number. Four bulls is a win. The player continues guessing until he succeeds or gives up.

The first move is complete pot-luck but subsequent moves can be calculated statistically from the information gained by previous moves. A good MOO player can average under 5 moves per game.

example:

```
.MOO<cr>

↑
1234<cr>

BC                              : one bull and one cow

↑
4567<cr>

NO BULLS OR COWS

↑
3891<cr>

BBCC

↑
9831<cr>

BBBB YOU TOOK 4 MOVES

EXIT
↑C

.
```

MOO commands:

Any 4 digit number with all digits different is a move
Type 'R' to obtain the rules.
Type '?' to give up. MOO will reveal its number.

LIBRARY ACCESSIONS

NEMES, Tihamér	*Cybernetic machines* (001.53028 NEM Main)
TARRANT, John Rex	*Computers in geography* 1970 (016.91 TAR Main Ref.)
INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS	IEEE publications bulletin v.1 1970 and onwards (016.6213 INS Elect.)
LOHNES, Paul R.	*Introduction to statistical procedures* 1968 (311.018 LOH Rem.Ed.)
DANIEL, Coldwell	*Mathematical models in microeconomics* 1970 (HB74.M3D294 Main)
BÜHLMANN, Hans	*Mathematical methods in risk theory* 1970 (368.00184 BUH Maths)
ZINN, Karl L.	*Comparative study of languages for programming interactive use of computers in instruction* 1969 (Qto 371.3944 ZIN Engin.)
OLSHEWSKY, Thomas M., ed.	*Problems in the philosophy of language* 1969 (P106.04 Main)
DANIEL, James W.	*Computation and theory in ordinary differential equations* 1970 (517.6 DAN Maths)
FRÖBERG, Carl Erik	*Introduction to numerical analysis* 1970 (517.6 FRO Elect.)
GHOSAL, A.	*Some aspects of queueing and storage systems* 1970 (519 GHO Maths)
INTERNATIONAL CONFERENCE ON COMPUTING METHODS IN OPTIMIZATION PROBLEMS 2d, San Remo, Italy 1968	*Computing methods in optimization problems* 1969 (517.6 INT Maths)
McDANIEL, Herman, ed.	*Applications of decision tables* 1970 (519 MACD Engin.)
SOCIETY FOR INDUSTRIAL AND APPLIED MATHEMATICS	SIAM journal on mathematical analysis 1970 and onwards (QA300.S825 Maths)
AHMED, F.R., ed.	*Crystallographic computing* 1970 (548.7 AHM Chem.)

BODDY, David E.	*Engineering design computation manual* 1969 (Qto 620 BOD Engin.)
HOLDEN, T.S., comp.	*DAD system reference manual* 1970 (Qto 621.381958 HOL Engin.)
IFAC SYMPOSIUM ON PULSE-RATE AND PULSE-NUMBER SIGNALS IN AUTOMATIC CONTROL, Budapest April 1968	*Proceedings* 1969 (Qto 629.8 IFA Elect.)
COLLINS, George O.	*A study of the remote use of computers* 1967 (Qto 658.84 COL Engin.)
CONFERENCE ON DATA SYSTEMS LANGUAGES. Systems Committee.	*A survey of generalized data base management systems* 1969 (Qto 651.84 CON Engin.)
	File organisations 1969
GAUTHIER, Richard L.	*Designing systems programs* 1970 (651.8 GAU Engin.)
INTERNATIONAL BUSINESS MACHINES CORPORATION	*Student text* 1969 (Qto 651.8 INT Arch.)
INTERNATIONAL SEMINAR ON ADVANCED PROGRAMMING SYSTEMS Jerusalem 1968	*Notes* 1968 (Qto QA76.5.152 Main)
McKEEMAN, William M.	*A compiler generator* 1970 (651.8 MACK Engin.)
SLUTZ, Donald Ray	*The flow graph schemata model of parallel computation* 1968 (651.8 SLU Engin.)
VAZSONYI, Andrew	*Problem solving by digital computers with PL/1 programming* 1970 (651.8 VAZ Engin.)
UYS, J.M., ed.	*Process simulation and control in iron and steelmaking* 1966 (669.10184 UYS Engin.)
MONTANARI, G. Ugo	*Separable graphs, planar graphs and web grammars* 1969 (Qto 744.4 MON Engin.)
BOTVINNIK, Mikhail Moiseevich	*Computers chess and long-range planning* (794.1018 BOT Maths.)
ASHTON, P.	*Systems analysis and design* 1970 (The 4166 Acc.)
COMPUTER AND INFORMATION SCIENCES SYMPOSIUM, 2d, Battelle Memorial Institute	*Computer and information sciences* 1967 (001.53 COM Engin.)
OTTESTAD, Per.	*Statistical models and their experimental application* 1970 (001.424 OTT Biol.)

ABSTRACTS of the theses approved for D.Sc., Ph.D., M. Tech. and M.Sc. degrees	and postgraduate diplomas v.1. 1955-1966 (013.37854 ABS Main)
NIE, Norman	*SPSS. statistical package for the social sciences* 1970 (Qto 029.7 NIE Engin.)
BARRETT, David Arthur	*Automatic inventory control techniques* 1969 (HD55.B34 Main)
INTERNATIONAL MANAGEMENT CONGRESS, 15th, Tokyo 1969	*Proceedings* 1969 (Qto HD29.165 Main)
MURPHY, Judith	*School scheduling by computer* 1964 (371.242 MUR Arch.)
AMERICAN LIBRARY ASSOCIATION Science and Technology Reference Services Committee	*A guide to a selection of computer-based science and technology reference services in the U.S.A.* 1969 (507.2073 AME Main)
UNITED STATES OF AMERICA STANDARDS INSTITUTE	*U.S.A. standard code of information interchange* 1968 (Qto Q370.U5 Main)
BOOT, Johannes Cornelius Gerardus	*Quadratic programming* 1964 (519.92 BOO Maths.)
KUNZI, Hans Paul	*Nonlinear programming* 1966 (519.92 KUN Maths.)
MAL'TSEV, Anatolii Ivanovich	*Algorithms and recursive functions* 1970 (511.8 MAL Maths.)
MARTIN, James Thomas	*The computerized society* 1970 (QA76.M36 Main)
MULLISH, Henry	*An introduction to computer programming* 1966 (519.92 MUL Maths.)
ASSOCIATION FOR COMPUTING MACHINERY	*Special Interest Committee on Symbolic and Algebraic Manipulation* SICPLAN bulletin no 8 1965-1967 (QA76.A78 Engin.)
ASSOCIATION FOR COMPUTING MACHINERY	*Special Interest Group on Symbolic and Algebraic Manipulation* SIGSAM bulletin no 9 1968 and onwards (QA76.A78 Engin.)
SCHOOL MATHEMATICS STUDY GROUP	*Algorithms, computation and mathematics. Teacher's commentary* 1966 (519.92 SCH Engin.)

COMMONWEALTH SCIENTIFIC AND INDUSTRIAL RESEARCH ORGANIZATION Division of Computing Research 3200 System Group	*User's manual for the 3200/3600 CSIRO computors* 1969 (Qto 621.38 1958 COM Engin.)
CROWLEY, Thomas H.	*Understanding computers* 1967 (621.38195 CRO Engin.)
DIEHL, Charles E.	*Survey of current practice in the use of automated techniques for specifications and detailing practices* 1967 (Qto 624.177 DIE Arch.)
HELLERMAN, Herbert	*Digital computer system principles* 1967 (621.38 1958 HEL Engin.)
ARTHUR YOUNG & COMPANY	*Computer auditing in the seventies* 1970 (Qto 657 ART Acc.)
COMPUTER science	1969 (651.8 COM Engin.)
COMPUTER science	*FORTRAN language* 1970 (651.8 COM Engin.)
CONFERENCE ON DATA SYSTEMS LANGUAGES Programming Language Committee Data Base Task Group	*Report* 1969 (651.84 CON Engin.)
DAVIS, Gordon Bitter	*Computer data processing* 1969 (651.8 DAV Engin.)
GREEN, Richard ELLIOT, ed.	*Computer graphics in management* 1970 (651.8 GRE Engin.)
NATIONAL ASSOCIATION OF AUSTRALIAN STATE ROAD AUTHORITIES	*Glossary of computer terms* 1970 (651.8 NAT Engin.)
ROSS, Joel E.	*Management by information system* 1970 (658.501 ROS Acc.)
STERLING, Theodor D.	*Computing and computer science* 1970 (651.8 STE Engin.)
WALKER, Terry M.	*An introduction to computer science and algorithmic processes* 1970 (651.8 WAL Engin.)
ARDEN, Elizabeth	*Systems analysis and design* 1970 (The 4167 Acc.)
BRAES, John Ratcliff	*A digital computer simulation of the flotation process and investigation of some possible control variables* 1970 (The 4187 Main)
LEES, M.J.	*Simulation techniques applied to industrial Comminution circuits* 1970 (The 4181 Main)

MATRIX INVERSION AND SOLUTION OF SIMULTANEOUS EQUATIONS

Ian Oliver

Many GE-225 and PDP-10 users are including highly unsatisfactory matrix inversion subroutines in their FORTRAN programs. One example is the subroutine published in Veldman [1]. When given the 2×2 matrix

$$\begin{array}{cc} 0.6 & 0.2 \\ 1.8 & 0.6 \end{array}$$

this subroutine produces the 'inverse'

$$\begin{array}{cc} 0.13E9 & -0.45E8 \\ -0.40E9 & 0.13E9 \end{array}$$

On inverting this 'inverse' the original matrix should be obtained. Instead the subroutine gives the result

$$\begin{array}{cc} 0.\dot{3} & 0.\dot{1} \\ 1.0 & 0.\dot{3} \end{array}$$

The original matrix is singular and in fact has no inverse at all. The subroutine should have detected this condition. However, there is no error indicator in the calling sequence of Veldman's subroutine which could be used to return any information about the success or otherwise of the inversion. What is particularly worrying is that the results from such a subroutine can sometimes look 'reasonable' even though they are completely wrong.

The reason for the erroneous calculation above is that the elements of the matrix are stored in binary floating point format which cannot represent those decimal values exactly. They will be correct only to about eight digits. Thus, although the matrix is exactly singular it does not appear so to the computer.

In practice matrices from experimental data are rarely precisely singular. On the other hand it is very common for them to be 'nearly' singular. This means that, in some sense, the determinant is very small. In the case of the 2×2 matrix A the determinant is given by

$$|A| = a_{11}a_{22} - a_{12}a_{21}$$

If this value is small relative to the size of the elements of A it is clear that the subtraction had operated on two very nearly equal quantities. Very few (if any) significant digits would be left and the result would be quite meaningless. Subroutines such as Veldman's give no indication of trouble. This is precisely what happened in the above example. The determinant of the matrix is exactly zero but was computed as about 0.7E-9.

We recommend the library subroutine MATINV (classification number D4.205 for the GE-225, and D4.505 for the PDP-10). This subroutine checks the result of every subtraction to ensure that the results will be correct within a tolerance value supplied by the user. It correctly determines that the matrix in the example above is singular.

Reference

1. Veldman, Donald J., *Fortran Programming for the Behavioural Sciences*, Holt, Rinehart and Winston, 1967.

USE OF DATA FILES BY FORTRAN PROGRAMS

Rob Cook

1. NAMING OF FORTRAN DATA FILES - ACTUAL AND DEFAULT

(a) Writing of Files

FORTRAN has 4 logical units available for disk I/O, namely 10,11,12 and 13. The data sent to any one of these unit numbers is written onto disk as a named file. The user can give this file any valid 5 character names he wishes by use of the OFILE Subroutine (ref. Bulletin Vol.4 No 2). However, if the program just WRITES to one of the above logical unit numbers without first calling the OFILE subroutine then the FORTRAN System will give the file the default name of FORn, when n is the logical unit number on which the WRITE was executed.

example:

Execution of the statement

```
WRITE (12) ARRAY
```

where there has been no earlier call to OFILE will create a file name

```
FOR12
```

(b) Reading of Files

Files can be read via the logical unit numbers. The user can access any named data file by first assigning the file name to the logical unit number with a call to the IFILE Subroutine (ref. Bulletin Vol.4 No 2). If no prior assignment is made, or if the named file does not exist, a READ is given on a logical unit number 'n', the FORTRAN system will attempt to find a file named FORn on the users area. If there is no file of this name the FORTRAN system makes a final attempt to find a file called FORTR. If this also fails the error message

```
?NO FILE FORTR ON DEVICE DSK
```

is produced.

(c) File Names

All file names for data files being accessed by FORTRAN programs must contain 5 characters or less, and have no processor program name. The 5 character restriction is imposed because the file name is held in the computer in ASCII in one machine word. All attempts to access a file where the name has six characters will therefore result in the above error message.

2. SIZE OF FILES

Any program running on the system may use up to 128K words of disk storage. The 128K words of disk storage may be allocated among the unit numbers in any way. The files that are accessed through each of the unit numbers are entirely independent of one another.

3. PHYSICAL AND LOGICAL RECORDS

A disk file is *physically* divided into records. Each physical record, called a *block*, contains 128 words. Thus a program's 128K words is subdivided into 1000 blocks.

A FORTRAN data file is *logically* divided into records. Each unformatted logical record contains the data referred to by an I/O statement. Each formatted logical record contains the data between a <cr> pair. A logical record is of variable length and may be longer or shorter than a block. It is important to understand the relationship between logical and physical records in order to pack data efficiently on the disk.

(a) Formatted Data

FORTRAN programs can read and write data files in two data modes. All formatted I/O statements use ASCII mode.

ASCII data is transferred character by character. ASCII logical records are packed into blocks, but are not split across blocks. Logical records must not be more than 128 words long. ASCII mode packs the maximum amount of data into a block.

Since formatted data is turned into ASCII characters before it is output, numbers are truncated to the numbers of digits allowed by the format field.

example:

Suppose A = 3.14159264 (single precision)

and is written to disk using the statement

```
         WRITE (1Ø,5Ø1)  A
   5Ø1   FORMAT (1H , F8.4)
```

and read back using a similar format; it would return as

```
         A  =  3.1416
```

(b) Unformatted Data

All unformatted I/O statements are binary mode. Binary data is transferred word by word. In binary mode FORTRAN reserves the first word of each block for its own use. This word is used to keep counts of

(i) the number of words used per block
(ii) the number of blocks per logical record.

Thus when writing in binary mode only 127 words per block are available. Since words are transferred direct in binary mode, no possible change in accuracy can occur.

Binary logical records are not packed into blocks, but they may continue over several blocks.

example:

Suppose A is a 100 × 100 array.

(i)
```
      WRITE (11)  A(1,1)
```
is very inefficient. It wastes 126 words in a block.

(ii)
```
      WRITE (11)  (A(I,1), I = 1,1ØØ)
```
is fairly efficient. It only wastes 26 words.

(iii)
```
      WRITE (11)  A
```
is very efficient. It uses $\frac{10,000}{127}$ blocks, and every word, in all but the last block, is used.

4. FILE PROCESSING

Under the current version of the FORTRAN operating system all access to disk files is sequential and in effect a disk file looks just like a magnetic tape file.

Most of the special FORTRAN statements that are designed for magnetic tapes, also work on disk files, for example

```
REWIND  n
ENDFILE n
```

BACKSPACE n currently works via Batch but not from Remote Terminals.

These statements are implemented for compatibility with magnetic tapes rather than to be of great use specifically for disk files. As with most attempts at compatibility in the computer world, this one does not really succeed. Owing to the differences in the characteristics of files on disk and magnetic tape, there are several combinations of I/O statement and special tape statements that cause errors.

example:

The sequence

```
READ (1Ø)    A
WRITE (1Ø)   B
READ (1Ø)    C
```

will cause the job to crash.

The safest way to use disk files on the current operating system is to use a

```
CALL IFILE (unit, name)
```

before a series of READS, or a

```
CALL OFILE (unit, name)
```

before a series of WRITES.