Loughborough University

This item was submitted to Loughborough's Research Repository by the author.
Items in Figshare are protected by copyright, with all rights reserved, unless otherwise indicated.

A task and methodology for studying man–computer interaction

PLEASE CITE THE PUBLISHED VERSION

PUBLISHER

Loughborough University of Technology

LICENCE

CC BY-NC 4.0

REPOSITORY RECORD

Ashton, Richard G.. 2020. “A Task and Methodology for Studying Man–computer Interaction”. Loughborough University. https://doi.org/10.26174/thesis.lboro.13084076.

LOUGHBOROUGH
UNIVERSITY OF TECHNOLOGY
LIBRARY

AUTHOR

ASHTON, R G

COPY NO. 057839/01

VOL NO. CLASS MARK

~~LL 30 '79~~

12 JAN 1979

DUE FOR RETURN

20 JAN 1979

LOAN 1 MTH + 2
UNLESS RECALLED

27 MAR 1979

LOAN COPY
DUE FOR RETURN

27 MAR 1979

LOAN 1 MTH + 2
UNLESS RECALLED

9 DEC 1988

17 MAR 1989

- 3 JUL 1992

~~3 JUL 1992~~

- 2 JUL 1993

- 1 JUL 1994

30 JUN 1995

~~25 FEB 1998~~

- 2 DEC 1999

005 7839 01

A TASK AND METHODOLOGY FOR STUDYING MAN COMPUTER INTERACTION

by RICHARD GEOFFREY ASHTON

A MASTERS DEGREE THESIS

Submitted in partial fulfilment of the requirements for the award of M.Sc. of the Loughborough University of Technology

October 1973.

Supervisor Professor B Shackel

Department of Ergonomics and Cybernetics

© by Richard Geoffrey Ashton

THESIS CONTENTS

ABSTRACT

Computer systems are reaching levels of effectiveness that fall short of their maximum potential. These shortcomings are due to a mis-match between man and computer and will remain until computer systems are designed with man acknowledged as the most important element in the system.

This thesis reports the findings of the first stage in a series of experiments designed to develop suitable tasks for, and techniques for measuring, man-computer-interaction. Once defined and validated these tasks can be used to study the effects of the computer's influence on user behaviour during interactive problem solving tasks. A simple resource allocation task is described as fulfilling the 3 main criteria of measurability, controlability and realism. The tasks are presented by a computer which also records the subjects performance data. Analysis of this data shows that, whilst there are large individual differences between subjects, the measurements are sensitive to small changes in human behaviour. The value of the task as a means of evaluating real world hardware and software is discussed, along with its use as a tool for further investigation into man-computer problem solving.

Loughborough University
Of Technology Library
Date Jan. 74
Class
Acc. No. 057839/01

W759 0720

ACKNOWLEDGEMENTS

In a thesis such as this many people give advice and encouragement. I must single out for thanks my sponsor, the Civil Service Department, my supervisor Professor B Shackel for his guidance and support, and my wife for her constant good nature. I also owe a great debt to the members of the Human Sciences and Advanced Technology Group of the Department of Ergonomics and Cybernetics, Loughborough University of Technology, mainly for their willingness to discuss my problems.

PART 1

LITERATURE REVIEW

PART 1 CONTENTS

1.1 Introduction

A study of the literature reveals an enormous gap between the highly sophisticated incredibly fast, stupid, computers of today, and our knowledge of the way humans solve problems using the computers as tools. In an attempt to explain why this terrific lag has developed we must look back to the beginnings of automatic computers, back to the days of Charles Babbage and his "difference engine" around 1822. Babbage anticipated the following characteristics of digital computer:

1. Punch card inputs
2. Arithmetic units or central processors
3. A memory
4. Some form of automatic print-out
5. Concepts of computer programming
6. Sequential program control
7. Automatic computer feed-back
8. 20 place accuracy

The development of digital computers then hit a lull during which desk calculators developed, and in 1890 Hollerith invented electrical techniques to read punch cards, resulting in the first electrical tabulating machines. The lull in the development of digital computer systems was mainly caused by the problem Babbage and others had of inadequate power supplies to drive long trains of gears. In 1892 Bush first used the electronic vacuum tube to provide amplification of the torque required to move these gears and pulleys. The next advance in digital computer development was World War II when, as is usual under the spur of war, science and technology accelerated at an unprecedented rate. Development of the vacuum tube saw the progressive displacement of mechanical parts by electro-mechanical and electronic components. The application of radar

feed-back in anti-aircraft control devices was an early and successful example of a computerized fire control system, widely used in World War II. A number of other computer controlled devices were in general use during this period but these computers were most often analogue devices. Analogue computers are deficient for various types of scientific or engineering problems; they cannot handle large numbers of variables, their accuracy is limited to 1 or 2 decimal places. They tend to be specific purpose machines with restricted applications, and they do not lend themselves to logical operations, program storage or self-checking with fast input and output. It was not possible to put Babbage's ideas of advanced digital computers into practice until the required electronic engineering technology became available.

The first digital computer was operational in 1946 called ENIAC. It was designed by Eckert and Maunchley in America, SACKMAN (1967). This was an enormous machine weighing 30 tons, using 150 kw of power and with approximately 20,000 valves. The first digital computers used wired programs and a big step forward in computer development was that of the stored program. From there computer operating speeds have progressed from the milliseconds speeds of the early computers to microseconds in the 50's to the present nanosecond speeds. High speed random access memory capability has been enlarged from 150 registers in 1946 to 10's of thousands in the 1950's and 100's of thousands in the 1960's. The 1970's are beginning to see millions of words of high speed storage and billions of words of auxilliary storage in the largest systems.

Programming systems have also advanced along with the advances in hardware technology. From the wiring techniques used at the end of World War II through machine language codes to the relatively machine-independent languages such as FORTRAN etc, right up to the problem oriented

languages tending towards natural languages that are being developed at this time. The widely held concept of an ideal natural ergonomic language has proved remarkably difficult to attain. Many problems such as linguistics, semantics and formal grammar have not yet been solved. It is here then that the human factors of computer software and interfaces must start. We have been left well behind by the rapid growth of computer technology. With the cost of computer time coming down and human time going up it is essential that man and computer interact more efficiently than ever before.

In 3 recent reviews of the area of man-computer interaction LICKLIDER (1965a), MILLS (1967) and DAVIS (1966) various views of this interaction have been taken but no cohesive underlying structure is apparent. For convenience the following review has been divided into 5 sections - Hardware, Language, Systems software, Communication and Interaction across the interface. This is adequate for a simple description of the "state of the art", but what is needed is a set of systematic relationships between the underlying dimensions contributing to the area of man-computer interaction. This taxonomy could possibly be on the same lines as that proposed by SHACKEL (1969), but it must recognise that taxonomies have a purpose other than allowing a structured description of the problem area. They must provide a set of relationships allowing the classification of research findings; to allow a coherent data base of the knowledge, gathered to date, to be presented. This is a complex problem as the aspects that contribute to man-computer interaction are many and it is difficult to describe multi-dimensional space.

1.2 Hardware interface

Perhaps the first thing any user sees on beginning to use the computer

system is the actual physical hardware of the system. I would suggest that there are 2 main aspects relating to hardware, the input hardware and the output hardware. The lowest complexity of input hardware that can be used in conjunction with the computer are simple toggle switches allowing the setting of absolute addresses within core memory and also allowing the setting of specific instructions such as JUMP, STORE and CLEAR. Input hardware extends from these switches through keyboards of varying complexity and graphical input devices right up to possibly the most complex input of users speech.

The second aspect to the hardware interface is the output hardware. The minimum possible hardware necessary to make the computer system function is a set of lamps indicating the state of the various bits of memory registers within the computer. Output hardware extends from these lamps through typewriter and print output and visual display devices to perhaps again the most complex output of computer generated speech.

These aspects of the hardware interface are probably the most easily defined and understood, consequently they have been the most researched aspect of man-computer interaction, from the human factors view point. One of the basic problems for the naive users in interacting with the highly sophisticated computers available today are the problems of getting information into and out of the computer in a manner that they can easily understand. On examining the data transmission methods of man we find he has 2 useful means of outputting data and information and 2 useful means of inputting to the computer from his own central processing unit. He can vocalize his instructions as an output, this is probably the fastest means of communicating between man and man but as yet computers are unable to understand human speech, in anything but a highly

simplified manner. The second means of conveying information is through a man's hands, which are capable of pushing keys, guiding pens etc. This is a much slower way of communicating with the machine or even other people. It has the advantage that it can be re-read at a later date; it has the disadvantage that skills unfamiliar to all men may be necessary, eg typing, card punching or keying. This may necessitate employing a third party, trained in certain motor skills.

As far as input to man is concerned the primarily medium is undoubtedly visual. Man's dynamic visual capability is far in excess of any mechanical dynamic visual device yet invented. By reading he can gain enormous quantities of information relatively easy. The other means of inputting to man is by his ears. Computers have already been programmed to produce speech but it will be a long time before this is generally available. When regarding the computer it would rather have information fed into it in serial or parallel binary form and would prefer to output information in serial or parallel binary form. Man has a very limited capability to understand serial binary code, as illustrated by the difficulty in learning morse and similar audio codes. There is also a vast difference in the rate of information output by the computer in serial/parallel form and the rate in which it can be understood, even by experienced morse code operators. The ratio for this means of information transmission may be as high as 1,000,000 to 1 in favour of the computer. Obviously then this means of information transmission is not suitable to either party in a man-computer interaction.

1.3 Language

The 3 terms - pragmatics, semantics and syntax - have been applied

almost indiscriminately in attempts to define computer language. In his original book "Foundation of the Theory of Signs" MORRIS (1938) defines the 3 terms as follows:

pragmatics - the study of the relation of signs to interpreters.

semantics - the study of the relation of signs to the objects to which the signs are applicable.

syntax - the study of the formal relation of signs to one another.

Generally it appears that within the framework of computer languages syntax describes the punctuation, grammar and allowable combinations of characters, semantics are the operations performed by the various synbols and pragmatics are the relationship between the language and its translator and its user.

To distinguish syntax and semantics we need to separate form and meaning. This is a particular problem when considering information processing because of its mathematical notation. Nowhere else have form and meaning been interfaced so intensively. As a matter of fact formalization is nothing more than packing as much meaning as possible into defined forms. To a certain degree this is done already in natural language, the plural is a syntactical form connected to a clearly established meaning. In our constructed languages we do the same thing in a much more elaborate fashion;the syntactic elements of constructed languages are an important part of meaning. Pragmatically this is very important because the human being when reading, has a kind of meaning assignment mechanism running, which will produce many errors if the artifically defined meaning is different from the usual natural one ZEMANEK (1966).

Since programming languages are a communication link between man and

machine we have 2 kinds of users of computer languages, an artificial one and a warm blooded one, a mechanical one and an illogical one. The first one is fully algorithmic and carries out what the text means to it while the second is heuristic and has notions, opinions and makes interpretations. This gives rise to 2 kinds of pragmatics, human and mechanical. The human factors engineer must concern himself with both types and not just the human side as both may directly affect the user.

Having described in general what any computer language consists of let us consider the more usual definition of computer languages and concentrate on high-level languages.

1.3.1. High level programming languages

SAMMET (1969) defines a high level programming language as a set of characters with rules for combining them and certain prescribed characteristics. Briefly these characteristics are:

1. That machine code is unnecessary for adequate use of the language. It is important however that the highly sophisticated user should be able to use the machine language if he requires, in order to make his program more efficient or more general or take advantage of peculiar machine characteristics, or even, if necessary, to manipulate the system to suit his own needs.
2. The language must be machine independent, having a potential for conversion to other computers.
3. The language must have a capability for instruction explosion. The language used will explode any program instruction into a simple machine code instruction, keeping

the original language relatively brief.

4. The language must have a notation that is closer to the specific problem being solved than normal machine code would be.

1.3.2 Advantages and disadvantages of high level languages

The biggest advantage claimed for a high level language is that is is much easier to learn than assembly languages. The second advantage of a high level language is that more attention can be paid by the user to the logic of the program, rather than to the idioscyncracis of the physical machine hardware which may be significant when dealing with assembly languages. Idiosyncracies such as ones's or two's complement arithmetic and conventions about positive and negative zeros can be awkward to use. Another advantage of high level languages is that they should be able to be learnt in small sub-sets, which is a very difficult thing to do with an assembly language. Because the notation of a high level language is more problem oriented, the actual program is generally easier to construct and understand than an assembly language program. A program written in a high level language is generally easier to de-bug than an assembly language program, firstly because there tends to be less code written and secondly because more attention can be paid to the logic of the program and less to worrying about the details of the assembled code. A program written in a high level language is also much easier to maintain and document than an assembly language program. A high level language program would also be easier to adapt for use on a different computer and in fact there is less chance of it needing adaption. Perhaps the greatest single advantage of a high level language is that it normally reduces the time from the inception of a problem to its solution.

These advantages do not however always exist. Assembly language may be a more economic way of solving a problem. The biggest disadvantage of high level language is that it requires additional compilation time which requires much more time than a straight assembly process. Another disadvantage of high level language is that the compilers always produce more object code for the same program than would be produced by an average programmer with an adequate assembly language. The program written in a high level language must be compiled many times which makes for a significant cost increase if the program is run many times. If the user does not know the machine code of the compiler or is unable to get object listings or core maps, and if the compiler does not provide adequate de-bugging diagnostics, a high level program may be more difficult to de-bug than an assembly language program.

A further disadvantage of a high level language is its inability to express all the operations that may be needed by a user.

Before discussing various types of languages and their structures it must be pointed out that although there are many types of language application there is one basic parameter that can influence any language. This is the interactive configuration of the entire computer system that the language is implemented on. The degree of interaction often gives rise to 3 sub-sets of the same language. Those designed for on-line systems, those designed for off-line systems and those designed for one type and modified for use on the other type. Those features of language that are dependent on the system are discussed in the relative sections later.

In the discussion on languages following I will not consider the system

aspects of language especially but consider language in general.

1.3.3 Classification of programming languages

There are various types of high level language and these can be broken down into the following sub-headings:

1. Procedure Oriented Languages

In this type of language the user specifies a set of executable instructions, which are performed in a sequence specified by the user. FORTRAN is an example of procedure oriented language.

2. Non-procedural Languages

There is some confusion as to the definition of non-procedural. It appears that the idea behind this class of language is that the user specifies the problem with executable operations, but that the user does not define the sequence in which these operations must be performed, this being done by a machine.

3. Problem Oriented Languages

A problem oriented language is best defined as a language in which it is easier to write programs to solve problems than an assembly language would be.

4. Application Oriented Languages

These languages are used for one single application such as machine, tool or process control.

5. Problem Defining Languages

These are languages in which the problem may be specifically defined as regards input/output but does not define the method of transformation.

6. Problem Describing Language

This is possibly the most ergonomic, in concept, of all the previous types of languages described. To use a problem describing language

it will only be necessary to make one statement such as "calculate pay roll for J Bloggs and Son Limited", which cites the problem in the most general way and which gives no instructions as to the method of solution leaving the computer to request data and solve the problem.

To complete the description of language SAMMET (1969) has suggested 4 purposes to which languages could be put and which must be specified before a language can be adequately described.

First and most important is the particular application area for which the language was designed. It must be determined whether it is to handle numerical, scientific or graphical displays, or business data processing, or engineering design etc. The second purpose is to specify the type of language, where in this context the following things are meant:

First, into which of the previously described classifications does the language fall?
Secondly, what is the form of the language relative to succinct and/or formal notations versus naturalness?
Thirdly, is the language meant for direct input to the computer or as a publication or reference language? and
Finally, what type of user is the language intended to assist?

1.3.4 The structure of programming languages

BENNETT (1969) and RAPHAEL (1966) have investigated the structure of programming languages and propose the language base can be simplified and the "Tower of Babel" problem removed. Any language can be said to

be made up of 5 basic elements.

1. The characters One of the first steps in describing a computer language is the definition of the usable character set. BENNETT proposes that each character should be assigned to a class by the person defining the language for the purpose of word delimitation. It must be possible however for the programmer to re-define easily any character that he wishes to use in his programme.
2. Words Words are composed of characters but are complete entities in themselves, each having its meaning exactly defined. A special class of word is called action operators. A word of this type causes an immediate action to be performed when it appears in a message string. All the verbs in high level languages are action operators as are most pseudo-operators of assembly languages.
3. Word Groups The grouping of words for various purposes is a process akin to quotation. It must be possible to group words within quotation marks, or open or closed parenthesis. This method permits unlimited recursive nesting of statement forms. In the more restricted environment where recursive nesting is not required, alternative rules can be established which do not involve parenthesis or other such quotation marks.
4. Lists These play a vital role in most computer languages and provide a natural and powerful method for describing the properties of language themselves. Lists are used in specifying the parameters of a given action operator, specifying the variables to be printed and giving the formal names of arguments when defining sub-routines etc.
5. Statement Forms There are 2 main types of statement forms; these are the infix and prefix forms. In the prefix forms the action word of the statements precedes the parameters that makes that action

specific. This form covers almost all statement types except the assignment statements of ALGOL and FORTRAN, where the statement appears as an algebraic equation. The equal sign or assignment symbol occurs between lists in which position it is called an infix form. For completeness it should also be noted that there is a suffix form which parameters appear first and the action operator last. BENNETT has suggested that if a language base is designed along these lines it would allow a unifying concept to be introduced so that it would not be necessary to have many different languages but only one which a programmer could modify and redefine at his convenience.

1.3.5 Factors influencing the choice of language

In 2 papers by CHAPIN (1965) and SCHWARTZ (1965) the problems of choosing a language were investigated. They suggest that there are 8 criteria by which to judge suitability of a language for any given application.

1. Personnel competence eg a person skilled in FORTRAN programming can often produce a workable program in FORTRAN regardless of the nature of the problem or other factors.

2. The nature of the problem When a problem can be expressed in mathematical notation an algebraic language such as ALGOL or FORTRAN is usually required. If the job requires only minimal amount of computation and large amounts of data arrangement COBOL or a symbolic manipulation language is usually chosen.

3. Ease of use and learning The high level languages are consistently easier to use than assembler languages. Less convenient are Report Program Generators, tabular decision languages and symbolic languages, because of the greater restrictions on the use mnemonics for identifying the operations to be done, or for the data to be operated on. For convience in learning, the high level

languages are preferred because less knowledge of the detailed working of the computer is required.

4. Speed of operation of the program Speed is worth money in most installations, especially when programmes are to be run repeatedly. High level languages have been criticised for wasting machine time. However it has yet to be proved that an average programmer using an assembly language will do any better in operation speed than an average programmer using one of the new high level languages, but the high level programmer should produce his program faster.

5. Speed of compilation and speed of programming Here again speed is worth money. Speed is the most important in installations with considerable volumes of new or revision programs. When programs are to be run repeatedly, compilation time can become an excessive burden. For simple programs that have a life of only a few weeks a high level language can be used to produce an answer much quicker than an assembly language program, and the addition of compilation time is insignificant.

6. Economy of storage Economy of storage can be gained at the cost of slower speed operation. The sought after balance is a combination of economy of storage and speed of operation. Control of this balance is most easily achieved in assembly level languages but only at the cost of considerable programming time.

7. The availability of Macro Commands These macro commands can make all the difference to a language and turn an indifferent language into an acceptable one. For example square root, logarithmic and input/output macro commands are features of some symbolic languages. The high level languages typically use macro commands extensively even to a point of being founded on them.

8. De-bugging aids This is possibly the most important aspect of

languages. The machine and symbolic languages do not often contain any sort of de-buggingaids. The high level languages usually supply extensive de-bugging facilities to the programmer to trap possible violations in the use of the language.

1.3.6 The use of English as a programming language

Periodically we hear arguments that we should use natural language, ie English, as a programming language. The reasons given for these proposals are usually, the high generality of natural language on the one hand and the easy availability on the other. Either reason is a misunderstanding of the purposes of computer languages. There are 4 reasons for the formalization of computer languages:

1. Security of Operation is assured.
2. Tacit pre-assumptions are excluded.
3. Notions are clarified.
4. Resolving structures can be applied to many other problems.

There are also 4 other reasons for the application of artificial languages:

1. Abstract notions are usually different from common ones.
2. The syntax of natural language is not exact.
3. Ambiguous words are eliminated.
4. Expressions become shorter.

It may be true that our ultimate meta-language must be English, but even if this is true it will be no reason to propose English as a programming language, or even as a first meta language. We do not want the full generality of natural language, as we wish our written programs to remain in the area that our science has already worked.

SAMMET (1966) when discussing this problem unfortunately included mathematical notation and other scientific notations under the heading of natural language. However this is not usually accepted within the definition of natural language and will not be accepted here. An average English sentence has about 75% redundancy, whereas mathematical or scientific notation has almost no redundancy. Nevertheless for certain applications such as question and answer systems and for quizzing data bases with a limited sub-set of natural language, a fairly good case can be made for using natural language. FRASER (1967) has suggested that with suitable definition of the syntax and semantics a sub-set of natural language can be used to quiz a data base about timetables, airline routes, connections, flight times etc.

BOBROW (1967) has written an interesting paper giving another view of the problems to be solved when constructing a computer question and answer system designed to interact with the computer in English. The system is viewed as containing 5 different parts; a parser, a semantic interpreter, an information storer, and an information retriever along with an English output generator. There is a need for extensive interaction between the sub-systems and the sub-systems and the user. The syntactic analysis described is based on a Chomsky type of transformational grammar. Semantic storage is characterised by a form of the predicate calculus, with additional algorithms for calculation, and structures designed for fast access to the relevant data.

1.3.7 Applications packages

Perhaps the biggest area of language yet to be developed is the specialised single-purpose applications languages. LICKLIDER (1965b) has described a system for designing application packages. He states that

before the application packages can become generally available, a coherent software data base must be implemented. He specifies an applications package language as having a number of attributes:

1. Since the user is often trying out procedures, the language must be procedure oriented as well as problem oriented.
2. The user needs to employ data structures and processes that he employed in the past or that were defined by colleagues, and he needs to refresh his understanding of those objects. The language must therefore have associated with it a meta-language and a retrieval system.
3. The user tends to construct his program piece-meal; after he has constructed several parts he must relate the elements together.
4. The user is willing to think in terms of a procedure construction long enough to build up a functional component of his model, but thereafter he merely wishes to activate it or cause something to act upon it or to have it contribute implicitly its program share to the system of which it is a part.
5. Although the user focuses attention on one aspect of the problem, and then another, of the process he is studying, he often expects the computer to maintain all the stipulated constraints and relationships at all times.
6. The user changes his mind frequently, he needs an on-line editing program that can operate on his substantive program, although this is not always true. This program should not be a part of the substantive program complex for he will use it in many different ways, it must therefore be part of the application language system.
7. Verbal graphic and alpha-meric inputs are inter-mixed. The user wishes to say words as he points to figures and he wants to write expressions with sub-scripts if there have to be sub-scripts,

but he would rather point to a highly sub-scripted variable than write it out.

8. The essence of an application language is its conversational "play-it-by-ear" nature. On average the conversation progresses but there are frequent instances of revision. Typically the user is running an incomplete program. The implementation part of the application language must therefore preclude attempts to execute parts of the program that do not as yet exist. It must provide temporary termination of uncompleted branches.

9. The users problem oriented thinking is hierarchial and the procedures implied by it are recursive. It is evident then that the applications languages have to be compared to procedural and graphic languages for their ability to handle hierarchial and recursive organisations.

1.3.8 Summary of languages

Some attention has been paid recently to the ergonomics of computers, but mostly concerned with layout and position of knobs and dials etc. I feel this effort is misdirected; a vehicle can be ergonomically perfect but unless it has fuel it will not go anywhere. Languages are the fuel of computers, we must get the engine working first and leave the cab design to later. The ergonomist and computer engineers must get together and design ergonomic software. Regardless of what type of languages are being produced it must have the following features:

1. There must be the minimum possible constriction on the users problem solving behaviour.

2. The language must be flexible; it must be capable of doing anything the user would wish it to do.

3. The language must be learnable in small sub-sets with a notation

as near as possible to the users preferred notation.

4. It should be suitable for all levels of user sophistication, so that the non-professional programmer can merely write procedures while the professional programmer can use the same language at assembly level.

5. The language should be totally machine independent.

6. The language should have a form which makes programs written in it easy to understand.

7. The language must be documented to a very high level.

8. The language should be capable of on or off-line use, or at least have 2 related sub-sets. When languages are generated having a majority of these features we may cease to see the language being the main constraint of the efficiency of the man/computer interactive process.

1.4 Systems software

Efficient operation of the system should not be dependent upon the user knowing the internal structure or details of the operation of the system or the service programs. The user should be free to do his thinking at the level of the language in which he and the computer are conversing. The system should require very little off-line training or instruction. Ideally it should be designed so that a novice can use it after spending a few minutes with a tutor or a manual and can expect to learn how to use it efficiently from feed-back provided by the system itself. As far as is possible the system should be designed so that the most efficient and powerful ways to a solution are discovered by the user as he progresses. That is to say the system should have a built-in teaching capability designed to facilitate the acquisition of that knowledge that will qualify the user as an expert. Unfortunately this

is not usually the case and the user has to struggle with an esoteric job control language with all their attendant problems of parameterization and default options. WEINBERG (1972) has suggested that parameterization of command functions will increase the speed of writing of statements by some 30% but that errors will increase some 3 times.

Where the system does impinge on the user it should do so with what can best be described as a convivial personality. BERKELEY (1969) in a fascinating article about the personality of an interactively programmed computer suggests a number of phrases that the computer should be prepared to accept, and produce in response to requests, or when it has a problem. The system is responsible for assisting the user whenever he gets into difficulties. There have been a number of articles describing such de-bugging aids - TEITLEMAN (1969), BERNSTEIN (1968) GRISHMAN (1970) and HALPERN (1965) have all suggested ways in which the de-bugging assistance given to the user by the computer can be improved.

Perhaps the most important aspect of this system is the influence it has on the users behaviour. In a series of papers NICKERSON, ELKIND and CARBONELL (1968),NICKERSON (1969) and NICKERSON and PEW (1971), NICKERSON proposed that there are 4 areas where the system may influence the users behaviour. He is concerned with how the system affects the users behaviour in such areas as system accessibility, responsiveness, predictability and charging policy. SCHERR (1966) noted that the effect of introducing a penalty on large programs was to reduce the average program length by one-third in the space of 3 months. Another aspect of the system that has been observed to influence user behaviour is

the overhead associated with getting on and off the computer. When the system has magnetic tape as its bulk storage medium the user usually experiences long delays in getting on and off the system. If however the system has a rapid access of storage capability the set up and tear down times are much shorter. This has the effect that with magnetic tape bulk storage systems the user plans ahead and uses long sessions, whereas when the penalty to get on and off the system is light he tends to come to the machine with a much less rigid idea of what he wishes to do. Another aspect of the system affecting user behaviour is response time, and this will be discussed more fully in the section on interaction.

Another system characteristic that has been observed to affect user behaviour is the charging algorithms. It has been noticed that if the user is charged simply on the basis of CPU time he tends to come to the console with the program barely formed, if however he is charged on a basis of on-line time he comes to the console with the program in a fairly complete state. It is possible by varying the relationship of these 2 aspects of the charging algorithm to mould the way in which the user will approach the system. However this is a very dangerous practice and not to be recommended, as it can lead to user dissatisfaction and rejection of the system.

1.5 Communication

The biggest problem in man/machine interaction is the flow of communication across the man/machine interface. Recently there has been a considerable upsurge in the use of graphics as a communication mode but there is almost no evidence by which to assess its value as a communication medium CHASEN and SEITZ (1967), COONS (1966). One of the biggest problems in the normal mode of communication is the verbosity of the computer to user

message. The experienced programmer/user gets very frustrated if he is forced to attend to long highly redundant messages, designed for naive users and the naive user is completely bewildered by the highly mnemonic messages desired by experienced users. Confounding this problem is the fact that an experienced user in one area may be a naive user in another area.

NICKERSON and PEW have suggested a number of ways this problem can be resolved.

1. There should be 2 separate operating systems. In one case the messages being complete and self-explanatory and designed for the novice and in the other being highly abbreviated and designed for the experienced user. Unfortunately this solution does not take care of the naive user who is also an expert in certain areas.

2. One program 2 message sets This solution provides the 2 message sets described previously in one program and allows the user to select which message set will be output. The problem now is to decide whether 2 message sets are sufficient to meet the demands of the complete novice and the very experienced user. Perhaps 4 or even 5 message sets would be preferable.

3. Interrupt button Another possibility is to provide the user with an interrupt button so that he can stop the message when he has had sufficient information. For this approach to be fully effective it should be possible for the user to terminate any message by pushing a key at any time during the message output. With this capability the user only need attend to the output as long as it is informative. This raises problems of the position of the critical information within any message.

4. Two-part messages It is possible to store each computer-to-user message in 2 parts. The first part, a highly abbreviated mnemonic form, always being output and the second part, the complete explanation, being output should the user request it. This has the distinct advantage that the naive user when requesting the fuller explanation will also see what mnemonic code was associated with it.

Perhaps the ideal answer is a combination of (3) and (4), the interrupt facility with a two-part message.

Another problem of communication is the format of messages. Some experiments have been conducted in this area, by ROOT & SADACCA (1967) and CORNEY and TAYLOR (1970). However there is still a great deal of work to be done before best input formats have been determined. Of equal importance is the output formats from the computer to the user. Some of the problems associated with this have been discussed by PARSONS (1970). He suggests that the output format for messages should be designed by the users rather than systems programmers. I am not sure however that this approach is entirely valid, as it may yield a plethora of output formats with little standardisation between them. However there is certainly a need for some research in this area. Another problem of communication is that of documentation. As most systems are not at the moment self-documented there is a considerable need for good documentation in order to make systems effective. A series of papers describing the documentation of significant current programming languages introduced by YNGVE and SAMMET (1963) would seem a reasonable point at which to start producing better documentation.

1.6 Interaction across the interface

Already in this thesis such words as time sharing and interaction have occurred. Interaction is basically concerned with the response time of the machine to the user. SCHWARTZ (1968) has written an excellent paper discussing the past, present and future possibilities of interactive systems. NICKERSON (1969), and CARBONELL et al (1968) have suggested that the system response time may significantly affect the users behaviour. Fast response time has generally been considered to be the *sine qua non* of interactive systems. Moreover the faster the better appears to be the philosophy. However, response times tend to vary with such things as load on the system and the command being executed. There are some indications that the variability of the response time distribution may be nearly as important from the users point of view as its mean. It appears that the degree to which the user is frustrated by a delay is not simply a function of its duration, rather it depends upon such psychological factors as the degree of uncertainty of the user concerning how long the delay will be and the extent to which the actual delay contradicts his expectations, and also what he considers to be the cause of the delay. That the user's uncertainty concerning how long he must wait for a response from the computer may be a greater source of aggravation than the delay itself was pointed out by NEISSER (1965). One implication of this is that a relatively long response time of constant and known duration may be in some cases more tolerable than a highly variable response time with a short mean.

SIMON (1966) has suggested that some thought should be given to quantitizing delays rather than allowing them to vary over a continuous range. The idea being that if the computer cannot respond immediately to a request the delay should be artificially extended to a known duration

so that the user can conveniently turn to another task. Unfortunately this raises considerable problems, as yet unsolved, as to the user's job swopping capability. We know virtually nothing about human job swopping behaviour. Do people have a minimum job swop time? How does swopping between tasks affect the performance on both tasks and on the combination? How does the time required to make an effective swopping depend on the nature of the task? Is swopping time quantitized or is it continuous function of task complexity? These problems can best be resolved by controlled experimentation.

MILLER (1968) has suggested that response time is related to other psychological phenomena such as the response expectancy. He suggests that when addressing another human being the individual requires some response within, say 2-4 seconds. Even though this response may have no information in it, a response is still expected within that time, it may be no more than a clearing of the throat or a grunt. He suggests that in conversations of any kind between humans silences of more than 4 seconds become embarrassing, as they imply a break in the thread of communication. He suggests that there is a second psychological need in communication. This need recognises that humans spontaneously organise their activities into clumps, characterised by the subjected completion of a purpose or sub-purpose. Psychologists call this subjective sense of completion, closure. The hypothesis is that more extensive delays may be made in a transaction after a closure than during one. Another psychological variable affecting response time is human short-term memory. In complex problem solving, short-term memory is very heavily used, and it is becoming clear from the psychological literature that the degree of complexity of a problem that can be solved is dependent upon how much information

can be held in short-term memory. MILLER suggests, for a number of types of interaction between the man and the machine, some minimum response times. However these times are based on opinion, albeit expert, and have not been experimentally validated.

1.7 Summary

The technology of computers has advanced very fast over the past few years without a complementary increase in our knowledge of the way people react to using computers as tools. Some research is being done aiming at trying to understand these complex relationships, but there is no classification system with which to relate this research. Without an adequate taxonomy the research done will continue to be fragmentary and disjointed.

The most researched area of man-computer interaction has been the hardware used by the programmers and computer users; however there are still many problems of data entry and output that still have to be solved, especially in the field of dynamic visual devices. Arguments still rage over the choice of devices such as teletypes or Visual Display units, over joysticks rolling balls and light pens for visual displays, and there appears to be no way to settle the arguments.

In the preceding literature review great attention has been paid to languages; this is because this area is the least researched from the human factors viewpoint and yet is probably the most important from the user's view. Statements regarding the power, ease of use, and ease of learning of languages are prevalent in the manufacturers' advertisement and as far as I can ascertain no measurements of these factors have been attempted. Objective measures of power, ease of use and suitability for

the purpose are needed and it must be the responsibility of the ergonomist to provide them.

The relationships of systems software to applications software needs the attention of ergonomists, as does the effect of the choice of system parameters on the user. At the moment we only have case study evidence of the effect of charging algorithms etc; there is a pressing need for controlled experimentation in this area. There are very many other problems, highlighted in the previous literature review that should command the earliest attention of ergonomists. It is not sufficient to complain about the difficulties of measurement and control; techniques must be found or else the computers will make slaves of its users and we will have another example of the adaptability of man and his capability to use the most un-ergonomic systems.

PART 2

EXPERIMENTAL DESIGN

PART 2 CONTENTS

2.1 Introduction

The problems outlined in Part 1 of this thesis have only a few things in common, but of paramount importance across all problems is the realisation that Man-Computer-Interaction is interaction in the full sense of the word: it should be ongoing, dynamic and changing.

The first step towards a solution to the specified problems is to establish some experimental procedure for evaluating variables impinging on any specific situation. To match the nature of the problem it is essential that the experimental situation has all the elements of interaction that are likely to be present in the real world.

Of the reported research, only 3 papers are relevant to this thesis. The earliest by GOLD (1967) was perhaps the first real attempt to evaluate man-computer-interaction in a realistic manner. Unfortunately the problem Gold set himself and the task he used to present the experimental variables were in my opinion far too complex for the current state of the human science knowledge at that time and were inexactly designed and controlled. Gold was concerned not with measuring interaction specifically but in trying to establish a difference in value and performance between time-sharing and batch processing for problem solving situations.

JONES, HUGHES and ENGVOLD (1970) in a similar study to Gold's were concerned with the problems of management decision making at computer terminals. They were interested in comparing the relative merits of typewriters and visual displays in resource allocation problems. They used the computer to assist the scheduler to produce a job-shop schedule for a small machine shop. The schedulers were required to select

decision rules and the computer would then generate a schedule which the scheduler could then 'fine tune' to produce the best possible profits. The computer would do all the profit, loss, price per hour and overtime calculations necessary. The experiment was run with 3 main conditions where the schedules were produced by manual means, a computer driven teletype and a computer driven visual display. They found no performance difference across computer groups on the measure of maximum profit; however the performance variance of the display group was lower than the variance of the teletype group. They also observed a large difference in the number of schedules generated. The display group generated more schedules than the typewriter group which in turn generated more schedules than the manual group. One conclusion they came to was that a visual display used in a scheduling problem raises the level of performance of the weak performers.

The third and most recent study by BOEHM, SEVEN and WATSON (1971) was concerned with a hypotheses suggested by Gold. They said, "that restricting access to the computer for a period of time after the presentation of current results - 'lockout period' - might improve performance by inducing the user to concentrate more on problem solving strategy than on tactics". They used a very sophisticated task which required the subject to utilise the computer to optimise the placing of hospitals in an area, taking into account relative accident densities, roads and freeways.

The subjects manipulated their problem with a specially constructed applications package by providing some decision rules and requesting the computer to evaluate these rules and display tables of numbers for further evaluation. It is perhaps a little harsh to suggest that

the only thing they found was that people are different, but they failed to prove or disprove the 'lockout' hypotheses.

These 3 experiments have failed in one way or another to live up to the expectations of their authors. This by no means invalidates the research but it does tend to suggest that they had been trying to do too much in one go. It is readily apparent from the literature that the computer at this moment, instead of producing much vaunted symbiotic relationships, is merely serving as an amplifier of individual differences producing very good or very bad systems. This being the case I suggest it is essential that a simple standardised task be produced, embodying all the elements present in an interactive situation, that can be applied to different user groups and user problems. It should provide measurements that can be correlated with psychological measures of human variability, thereby producing the means for evaluation of hardware, software and man for any given system.

The ultimate purpose of this research is to produce such a task along with the means of measuring all the relevant man and computer variables in a given task situation. This thesis however is concerned with the first stages of such a research program: to define, try out and modify a suitable task.

2.2 Tasks

2.2.1 Task requirements

The general requirements of such a research vehicle are:

Measurability

Controlability

Realism

1. Measurability

In such a man-computer situation it must be possible to measure objectively all the input to and output from the man. When experimenting with man-computer-interaction one has at least the facility to record vast quantities of numbers. However without the means and hypotheses to analyse these numbers and relate them to real world variables there is no real point in recording lots of numbers. This means that measurability must include the ability to analyse.

2. Controllability

It must be possible within the task situation to hold all identifiable variables constant, except of course the experimental variables. The prime requirement in order to control variables is that they be identified; once identified they are fairly easy to control.

3. Realism

There are 3 aspects to realism. The task must appear realistic to the subject when it is compared with the real world (verisimilitude), and it must be realistic from the experimenters point of view; this is where many previous tasks have failed in attempting to be too comprehensive or subjectively realistic. Thirdly, it must be realistic from a scientific viewpoint; there is no point in having subjects and experimenters happy and then producing analysed results having no significant relation to the real world (fidelity).

There are a number of lesser requirements that are nevertheless important:

1. Validity

The task must be valid so that we are testing and controlling

what we think we are and not some artifact of the task-computer combination.

2. Task sophistication

The task must be simple for the subjects to grasp, but allow experienced subjects to use nuances of strategy or tactics, and these variations should show as real performance differences. Only if this is possible will a better understanding of man-computer problem solving become possible.

3. Motivation

The task must be motivating to the subjects. They must get engrossed in what they are doing so that the interaction flows smoothly. This cannot happen if they are trying to beat a system they resent.

4. Goals

The subjects must be given the same goals and assessment criteria, and these must be clear and unambiguous and what the experimenter desired.

5. Response time

This important consideration will be treated in detail later, Section 2.3.10.

2.2.2 Preliminary task specification

Having decided on the general properties of such a task the next stage is to specify and design it exactly. There are a number of different tasks that fulfil some of the above general requirements. These are - text editing, a manipulation type task, man-man or man-machine games, some adapted standard psychological tests, and resource allocation or resource scheduling tasks. From an analysis of the alternatives I believe the type of task having the least number of disadvantages for this research vehicle is the resource allocation type.

There is, in most experimental situations, a further restriction on the design of a general purpose task. This is the availability of hardware and software on which to implement the task. Before going on to describe the tasks used in these experiments a brief description of the hardware available to the author is necessary.

2.2.3 Hardware availability

The Department of Ergonomics and Cybernetics at Loughborough University of Technology possessed in 1972 a Digital Equipment Corporation PDP-12 computer. This is a small extremely versatile Laboratory Instrument/ Data Processor computer, ideally suited for man-computer-interaction research. The configuration used in these experiments was as follows. 8K of core supported by 2-132K magnetic tape drives and an ASR33 teletype. The PDP-12 also has a real time clock which was used, but the Visual Display Unit could not be utilised due to insufficient core. All the software for the tasks and analysis, on both the department's PDP-12 and the university's ICL 1904A, was written by the author with the single exception of the Analysis of Variance Program.

2.2.4 Resource allocation task

The only task that fulfils every general requirement to some degree is the resource allocation type of task, as used by Gold, Jones and Boehm. Having decided that the subjects will interact with a resource allocation program there is still another decision to be made. In man-computer-interaction the user can basically do 2 things, he can command the computer to perform some operation or he can enter data on which operations, that may also be specified by him, are performed. As suggested in Part 1 there are numerous problems in formatting commands or data, and so it was decided for this thesis to use only command

entry in the task, never requiring subjects to enter data.

The task used in these experiments was set as an electrical power station distribution problem. The subject was required to manipulate the system in order to optimise the loading on a supply. He had 7 areas each of which demanded a portion of the supply, the sum of the demands of all areas was always more than the supply available. The subjects task was to connect and disconnect areas from the supply in order to ensure the total demanded load matched the supply with as little supply as possible spare, but not overloading it. When the subject was satisfied with his optimisation he instructed the computer to run; the computer then informed him how effective his allocation was and changed one element in the situation for the next optimisation. The subject was required to do 30 of these optimisations in one session. He manipulated the system with a small command set which he typed on the teletype. There were 5 types of commands:

1. Listing Commands

There were 3 listing commands which when entered would cause the program to list some part of the present situation. The commands were:

a. DEMAND When this command was entered the program would list the existing demand of each area.

b. STATUS When this command was entered the program would list the existing status (ON, OFF) of each area.

c. SUPPLY When this command was entered the program would list the existing supply value.

2. Action Commands

There were 2 action commands:

a. ON When this command was entered the program would connect all 7 areas to the supply.

b. OFF When this command was entered the program would disconnect all the 7 areas from the supply.

3. Feedback Command

There was one feedback command:

TOTAL which when entered would cause the program to sum the demand values of those areas whose status was ON and output the sum to the teletype.

4. Arithmetic Command

There was one command the subject could use to assist with the task arithmetic. ADD which would cause the program to add the values of all the demands irrespective of status, and type the value on the teletype.

5. Operational Command

This command GO would cause the program to evaluate the latest alterations, compute and output an error score, and then change one element of the situation.

All the above commands acted on all 7 areas and it was therefore necessary to specify that the command should apply only to some selected areas. This was achieved by the subject typing the number of the area he wished the command to be executed for after the command. This parameterisation is ineffective for the commands SUPPLY, TOTAL or GO as these do not involve areas. The default condition, if no parameter string is supplied, is to execute for all areas.

2.3 Preliminary Experiments

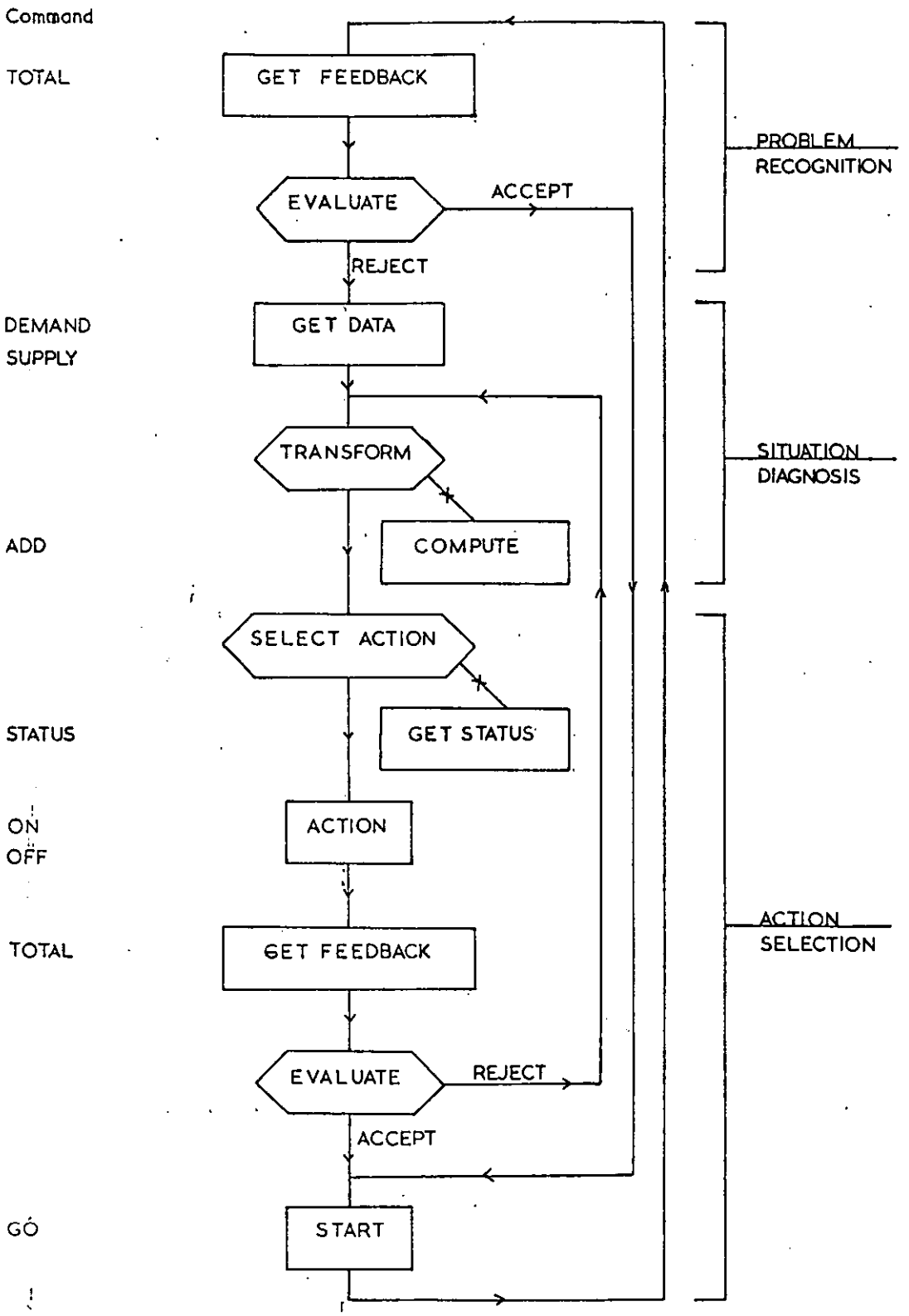
2.3.1 Pilot Experiment

With the task in this form a pilot experiment was carried out to see whether the whole approach was practicable and to test the task software

and the ability of subjects to manipulate the task. Six subjects were given a little instruction in the principle of operation and left unsupervised for 30 minutes. Before these experiments were started a theoretical solution strategy was constructed as shown in Fig 1.

It was suggested that the subject would first look at the TOTAL to gain information feedback about the correct position. If this was not satisfactory he would get information from the system in terms of supply and demand. He would then attempt to transform this information, possibly using the ADD command; when the transformation was successful he would select areas to be turned ON and OFF to implement his new optimisation. He may or may not use the STATUS command to gain information about existing status. He would then check with another TOTAL command and if satisfactory instruct the computer to GO.

This fits very well with the 3 main categories in man-computer-decision making suggested by SCHRENK (1969), these 3 categories being - problem recognition, situation diagnosis and action selection. All the subjects managed to grasp what was required of them without too much trouble. Settling down to a steady state after 2 or 3 cycles. However their actions did not appear to categorise neatly into the suggested strategic patterns. The implication of this will be consideredlater when the main experiment is discussed. Other observations of the pilot experiments were that subjects naive to computer terminals tended to forget to terminate their command line with the necessary 'return' key. They tended to wait, expecting the computer instinctively to know when to run, after they had finished their action selection. The command decoder was found to have excellent performance but the task was judged too easy and not realistic enough to represent a Management Information type of system.



PILOT EXPERIMENT THEORETICAL MODEL OF SOLUTION ALGORITHM

Fig 1

2.3.2 Main variables influencing task

Before re-defining and re-designing the task in the light of the pilot experiment experience, it was necessary to attempt to isolate the main variables influencing this man-computer-interaction.

1. Time

Perhaps the most fundamental variable in man-computer-interaction is time. There are 2 interactive modes of communication having a different time emphasis. These are Real Time operations where events are happening in real world time such as production control, process control, ticket reservation etc. It is also possible to be simply on-line to a computer with no real time pressures inherent in the task and only with a need to complete the task. It is possible to study both these situations with very similar tasks modified from the one previously described.

2. Feedback of performance

There are a number of possibilities in feeding back some performance measure or score to the subject. These are:

a. Not supplying any performance feedback at all.

b. Supplying error scores when requested.

c. Outputting error scores at fixed intervals.

d. Under overload conditions giving a different overload signal.

Within the task there are 2 levels of feedback, a micro feedback loop around the command decoder, and task performance feedback. The command decoder is possible as the teletype works in full duplex. (This means that separate send and receive software is used.) The action of the command decoder is discussed fully in Section 2.4.2.5.To feedback performance measures one of the problems is what the measure should be based on. There are a number

of possible measures which could be output when the subject types an Error or Score type of command.

a. A cumulative difference score of loads ON subtracted from supply, or

b. A points score based on the best fit possible and actual fit achieved.

3. Information to subject about change

In the pilot experiments the subject is always aware when the system has changed because it does so only when he typed GO. This condition will in future be called Subject Initiated Change (SIC). However if this task were used in a real time environment then a time base would be used to initiate a change (TIC), and the subject would not know when some parameter has changed without being told. There are 4 possible levels of change feedback:

a. No information about change.

b. An audible signal when change occurs.

c. An output statement as to what has changed.

d. An output statement as to what has changed and to what value.

4. Message Length

An extremely important variable in man-computer-interaction is the effect of various message lengths. As stated in Part 1 there are a number of possible ways of arranging information. The messages used in pilot experiments are stored in small pieces and are constructed into complete messages as required by the program. The teletype has a maximum capability of 72 characters per line. The program is capable of outputting messages of any length and any number of lines.

5. Ratio of change interval to message output time

In the TIC task the ratio of change interval to message output

time is of considerable importance. The message output time is dependent on the length of the message and speed of the output device. The teletype has a maximum rate of 10 characters per second. To output all the DEMAND information requires 17.8 seconds. To output all the STATUS information requires 14.7 seconds and to output the SUPPLY information requires 2.0 seconds giving a total time for listing outputs of 33.5 seconds, this is a maximum time using non-parameterised commands and no interrupts. If a ratio of 1:2 between message output and change interval is used, the changes would occur at 60 second intervals. This means there is no variability in the change interval. Another variable in the area of message length is the position within the message of the desired information. It would be very nice if it was possible to weight the message and its information position in terms of noise and redundancy. Unfortunately because each message may mean different things to different people such an objective measure is not possible. Because the messages used in the pilot experiment are redundant there is the possibility that subjects may wish to interrupt the output of a message after he has the information required.

6. Complexity of task arithmetic

The complexity of the task arithmetic is influenced by the short term memory ability of the man and his numeracy. The task is not intended to measure short term memory, so to keep the task's load on the subject's short term memory to a minimum, subjects must have a numerical ability suitable to the task. To reduce the load on mental arithmetic ability the value of the supply is limited to 3 digits and the values of the demands to 2 digits. In the pilot experiment all the values were exactly divisible by 5, but this

was judged to be too easy.

7. Choice of command words

In the pilot task the words were chosen intentionally to represent the action of the commands during execution. The command decoder can uniquely identify the first 4 letters of any command. For any command only the first 2 letters need to be entered if they are unique within the command set. If they are not unique the computer will output 'MORE' and at least all the first 4 letters must be entered.

8. Parameterisation

In the task the action of any command word is modified by the addition of a parameter string of up to 7 numbers specifying which area the command is to be applied to. Some recent work by WEINBERG (1972) suggests that using parameterised commands instead of single commands will increase speed of operation by 30% and increase errors 3 times.

9. Procedural instructions

In such a complicated task environment the instructions given to the subject are very important. It is possible to generate the desired output by accenting certain parts of the instructions. In the pilot experiment the instructions could not be included in the program because of core restrictions, but if possible this would eliminate some of the statements in the printed and verbal instructions.

10. Response time to input

The PDP-12 used in the pilot experiment is, as far as the subject is concerned, absolutely instantaneous in its response to an input command. This compares favourably with response times of other experiments reported, where response times have often been a

function of load on a time-sharing system. Response time usually means to the start of output. However, although rarely considered the time to completion of output could also be interpreted as response time, in which case the response time of 17.8 seconds to complete 'demand' listing output is outside the 15 seconds recommended by MILLER (1968).

2.4 Main Experiments

2.4.1 Selection of variables for further experimentation

After giving careful consideration to the importance of the foregoing variables it was decided that 2 of the most important variables need isolating before further research can be done using this task. These variables are the effect of information about changes on task performance in the TIC task, and the time relationship between task and real time. This led to the design of 2 tasks, one where the subject initiated the change as modified from the pilot experiment task and the other where the computer initiated the change at fixed intervals. To ensure the 2 tasks were as nearly as possible identical the GO command of the pilot experiment was removed and its action added to that of the feedback command, such that when the subject demanded feedback in the SIC task one value changed. The TOTAL command which was the feedback command on the pilot experiment was slightly modified and its name changed to SCORE. When the subjects type SCORE the computer output the difference between the sum of the demands of those areas whose status is ON and the supply, with a message indicating the direction of the difference, ie OVERLOADED or UNDERLOADED. The software was completely redesigned and a brief outline of its action is given here.

2.4.2 Task software

The complete software programs used in these experiments are included in Annex A. There are 7 parts in each of the task programs:

1. Start up

The start program is called by the main program. Its function is to set up the experimental variables and prepare the main program for output. It zeros all the data and sets constants, it requests from the experimenter the name of the message set and loads it into memory. It also requests the name of the task data set and reads that into the memory. It requests the name of the file to which data is to be output on completion of the experiment and opens it. It requests data regarding the subject his group number, trial number etc. It requests the number of changes to be allowed before it stops, it then starts the real time clock used to time event intervals. It waits for a command from the experimenter before passing control to the main program.

2. Main program

The main program checks the various data array indices for overflow and calls the command decoder. When control is returned from the command decoder it jumps to the appropriate action section for the command given, it does the necessary calculation and switching action to implement the command. It controls the call of alphabetic or numeric write sections of the program. When a particular command has been completed it recalls the command decoder.

3. Message output

This section receives a number from the main program looks up the appropriate part message and types it onto the output device returning control to the main program.

4. Numeric output

This section receives an integer number from the main program and converts it to a left adjusted decimal number with leading zeros suppressed. It types this and returns control to the main program.

5. Command decoder

This is the heart of the task software and is the only part that the subject has any real interaction with. The functions of the command decoder are as follows:

a. To accept from and echo to the typewriter certain acceptable characters.

b. To time input events and periods.

c. To verify input as syntactically legal.

d. To receive parameter strings.

On entry from the main program the entry time is written to a data array it then waits for the first character to be typed which it times from entry and stores in the data array. It accepts and echoes up to 19 characters but not all possible characters are legal.

It will accept only letters between A and Z, numbers between 1-7 and some special characters, space, comma, 'RETURN', 'RUBOUT', and 'ALT MODE'. If an input character is not legal it will not echo it.

When the input terminating character 'RETURN' is entered it will then check the input string for legality. It checks the first 2 input characters against an internal list, if these 2 characters are unique it notes the number associated with it and exits.
If however the first 2 characters input do not occur in the list it types 'EH?' and returns to the start. If the special character

'RUBOUT' is entered it will delete the last input character echoing a '≠'. If the special character 'ALT MODE' is entered it will delete all previous input since the last 'RETURN'.

As it exits, having accepted the input, it records the time in the data array and gathers the parameter information into a 'word' and stores it along with identification of the command entered. It then exits and returns control to the main program giving the number of the command and parameters entered. It is possible, after the command word has been entered, to separate the parameters, representing areas to be acted on, with spaces or commas. This command decoder system was never corrupted in many hundreds of command entries.

6. Timing unit

It is here that the main difference between Subject Initiated Change task and Time Base Initiated Change occurs. The internal real time clock is checked whenever the computer is waiting for its I/O peripheral in either the command decoder or message outputting systems. When it sees an interval of 100ms has occurred since it last operated, it counts that occurrence into a counter. On both tasks at intervals of 10 seconds the timing unit reads the value of those areas on at that time and stores their value in a data array. In the TIC task every 6 of these intervals it initiates a change. In the SIC task the same change system is used but only when the subject types 'SCORE' regardless of time.

7. Interrupt handler

When the program is outputting messages with its message output routines, before it types each character the interrupt handler checks to see if the keyboard has been struck, if it has the

interrupting character is cleared and output immediately halted. The fact that an interrupt has occurred is recorded as part of the information record of the next command. A typical section of interaction is shown in Figs 2 and 3.

2.4.3 Task complexity

One of the reasons for redesigning the task after the pilot experiment was that in the pilot experiment the changes occurred in pseudo-random fashion, both in what changed and its value, and it was obvious that more control over these changes was necessary. There are 2 variables that make up task complexity in this task environment. One is how well it is possible to fit the demands to the supply, the other is how often the combination of demands giving the best fit, and hopefully chosen by the subject, is changed.

The task data was therefore planned to have a certain difficulty. For the first 5 changes it was possible to get a perfect fit with the supply; during the next 7 changes it was not possible to get a perfect fit. During the next 9 changes it was again possible to get a perfect fit, and for the last 9 changes a perfect fit was again impossible. Coupled with this, during the first 5 changes the perfect fit was only disturbed once; during the next 7 the best imperfect fit was changed 5 times; during the next 9 changes the perfect fit did not change at all, giving 9 changes where no action was required of the subject at all if he got it right at entry; during the last 9 the best fit was changed virtually every time. This means that the task was easy to start with and got progressively more difficult. There was then a long flat period of very easy manipulation followed by the most difficult period.

```
*DEMAND

DEMAND AREA   1   27 MW.
DEMAND AREA   2   58 MW.
DEMAND AREA   3   65 MW.
DEMAND AREA   4   38 MW.
DEMAND AREA   5   26 MW.
DEMAND AREA   6   92 MW.
DEMAND AREA   7   73 MW.
*SUPPLY

SUPPLY IS  215 MW.
*ADD 123

ADDITION IS  150 MW.
*ADD 236

ADDITION IS  215 MW.
*ON 2,3,6

*SCORE

ERROR IS     0 MW.
*SCORE

ERROR IS     0 MW.
*DE

DEMAND AREA   1   50 MW.
DEMAND AREA   2   58 MW.
DEMAND AREA   3   65 MW.
DEMAND AREA   4   38 MW.
DEMAND AREA   5
*SC

ERROR IS     0 MW.
*SC

ERROR IS   10 MW. OVERLOADED
*OF 2

*SC

ERROR IS   48 MW. UNDERLOADED
*ADD 157

ADDITION IS  149 MW.
*ADD 167

ADDITION IS  215 MW.
*OF 3

*ON 17

*SC

ERROR IS     0 MW.
```

Section of a typical interaction Fig 2

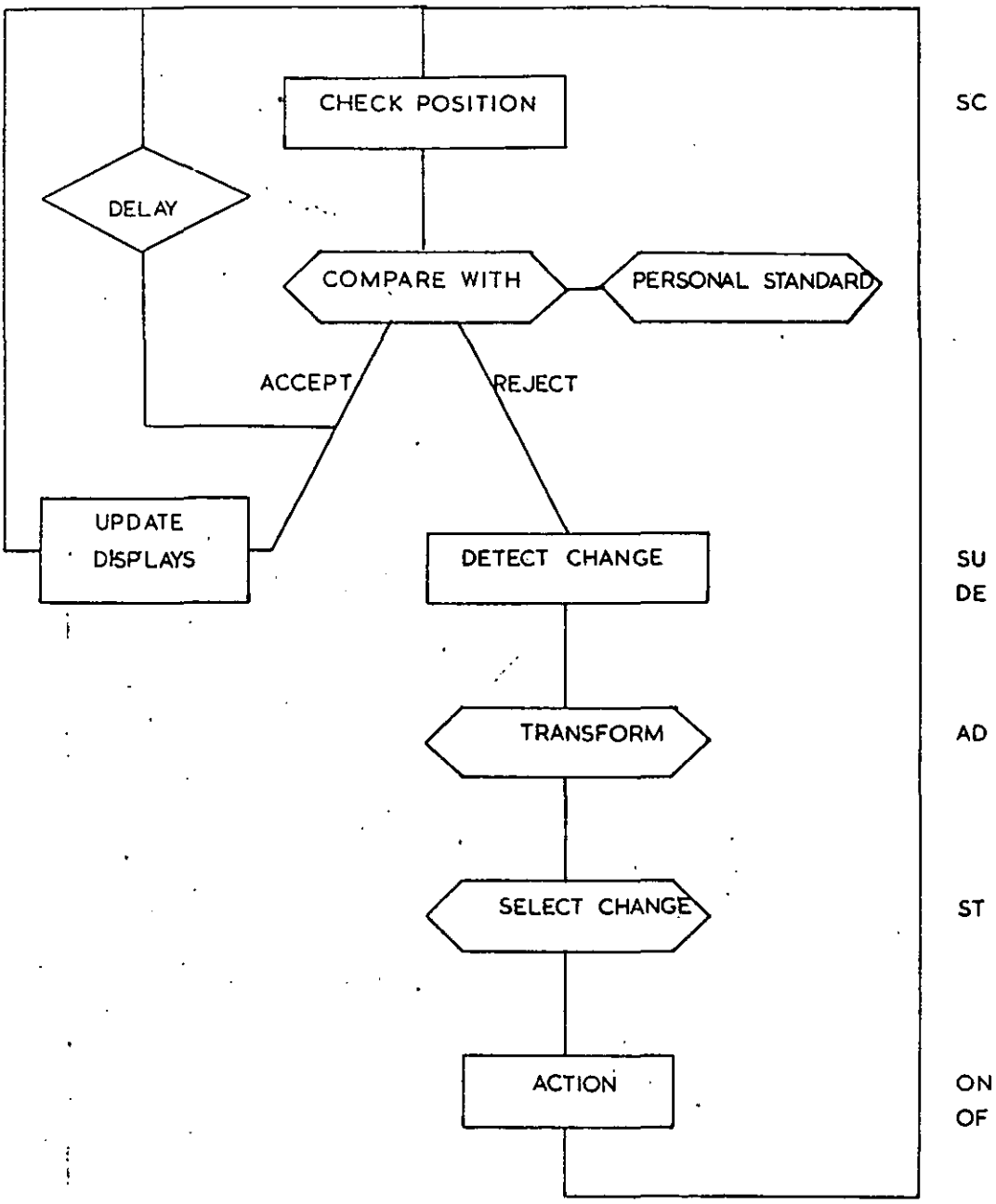
```
*ST

STATUS AREA    1 ON
STATUS AREA    2 OFF
STATUS AREA    3 OFF
STATUS AREA    4 OFF
STATUS AREA    5 OFF
STATUS AREA    6 ON
STATUS AREA    7 ON
*SC

ERROR IS      0 MW.
*DE

DEMAND AREA    1   50 MW.
DEMAND AREA    2   58 MW.
DEMAND AREA    3   75 MW.
DEMAND AREA    4   38 MW.
DEMAND AREA    5   84 MW.
DEMAND AREA    6   92 MW.
DEMAND AREA    7   73 MW.
*SU

SUPPLY IS   215 MW.
*SC

ERROR IS    13 MW. UNDERLOADED
*DE

DEMAND AREA    1   50 MW.
DEMAND AREA    2   58 MW.
DEMAND AREA    3   75 MW.
DEMAND AREA    4   38 MW.
DEMAND AREA    5   84 MW.
DEMAND AREA    6   79 MW.
D
*ADD 247

ADDITION IS   169 MW.
*ADD 257

ADDITION IS   215 MW.
*OF 16

*ON 25

*SC

ERROR IS
YOU HAVE FINISHED. THANK YOU VERY MUCH, GOODBYE!
```

Section of a typical interaction Fig 3

The task was designed so that 3 areas on would always produce the best solution though there were cases where 4 areas was as good but never better than 3. The range of numbers in the task was restricted to 2 digits for any area and 3 digit numbers for the supply. The distribution of changes was designed to be apparently random but each area or the supply had almost equal probability of changing, p = 0.125 for each area or supply over the whole task. It was necessary to have 2 sets of numbers to satisfy the experimental design. This was achieved by producing one set of numbers to fit the above complexity requirements and then generating a different set by subtracing a constant from the first set maintaining exactly the same pattern of changes, so that the same changes occur at the same times and the same areas were required ON for the same fit in both sets of task data. It was expected that subjects would not notice this fact but care was taken to avoid obvious sequences which would indicate similarity.

2.4.4 Possible solution algorithms

The solution model proposed for the pilot experiment was retained unmodified for the SIC task in order to gather more data before evaluating it. Another model solution based on the pilot experiment model but modified to incorporate a time base was proposed for the TIC task and is presented in Fig 4. Apart from testing these hypothetical solutions models some other results were expected. It was expected that the TIC task would show differences between situations where change information was or was not given. It was expected that more use of the command 'SCORE' would occur within the no-change feedback task, because it was hypothesised that the command 'SCORE' would be used to detect a change when its occurrence was not indicated by the computer. The SIC task compared with the TIC task was expected to produce



OUTLINE MODEL OF SOLUTION ALGORITHM

FOR T.I.C. TASK

Fig 4

differences in the learning effects in that subjects would perform better at the beginning on the SIC task, as they could take their time but would get pushed for time during the later stages, whereas in the TIC task they would be expected to be worse to begin with and to get better as time went on.

2.4.5 Experimental design

Two experiments were planned to run concurrently.

Experiment 1

To determine the effect of information concerning the occurrence of a change on task performance. The task used was the TIC task previously described. A bell rang at each change in one task (Bell), in the other task no indication of a change occurring was given (No Bell). Each of 8 subjects completed one,2-trial experimental period. Half the subjects were given the No Bell task on the first trial and the Bell task on the second trial (Group 1) and the other half were given the Bell task on the first trial and the No Bell task on the second trial (Group 2).

Experiment 2

To study the effects of a different time base on identical tasks. Each of 8 subjects, not the same as Experiment 1, completed one 2-trial experimental period. Half the subjects were given the TIC No Bell task on the first trial and the SIC task previously described on the second trial (Group 3). The other half of the subjects were given the SIC task on the first trial and the TIC No Bell task on the second trial (Group 4). In all 16 subjects were used in the 2 experiments. The 2 change data sets were alternated between task and experiments. The subject were all volunteers and qualified to 2 years under-graduate level or

better; their age range was 20-30 years and all were connected in some way with the Department of Ergonomics and Cybernetics at Loughborough University.

The subjects on arrival were given the appropriate instructions sheet (Annex B) and allowed to read them their questions were answered and when they were happy the program was allowed to run. After a few minutes to ensure they had no problems with the command decoder, they were left alone until the computer terminated the trial. They were then allowed a rest of 4-5 minutes whilst the task was changed over by the experimenter during which time they read the second instruction sheet. They then completed the second trial. Before the experiment started they were given a pre-test questionnaire and after this were given a post-test questionnaire. The subjects were then de-briefed and the purpose of the experiments explained to them.

PART 3

MEASUREMENT

PART 3 CONTENTS

3.1 Measurement

As previously described the command decode stores, as data, time measurements relating to input event times; in fact it stores the time of entry to the routine until the first character of the command was input, and the time from first character to terminating character. It also stores the command entered and which parameters were entered after the command as a separate data item. It stores the value of those areas ON, sampled every 10 seconds, during the tasks. As a preliminary analysis a program was written to convert this data into a more reasonable form. The time scores were converted, for each command, to a 'Wait' time calculated as the time from entry to first character, an 'Input' time calculated from first character to terminating character and an 'Output' time from termination of command to next entry. The Output time representing how long the computer took to execute the command whilst the Wait and Input times are subject dependent. Using this recorded data 3 main analyses were carried out. An Analysis of Variance on the Input and Wait times, a Command Sequence Analysis on the commands used, and a Cumulative Performance Assessment based on the error scores.

3.2 Analysis of Variance

The 2 experiments were analysed quite separately using this technique, WINER (1962). The Wait time and Input time were analysed separately and then combined to form a total input times for a third analysis. The factors of the Analysis of Variance were Subject (S) and Task Order (A). The factor A, Task Order, is nested within subjects as not every subject did every order. The other main factors were Task (B), Commands (C), and Periods (D). The Period factor was obtained by taking from the raw data the mean Input or Wait time for any particular command over 6 periods

of 5 minutes each throughout the duration of the task. The data was converted to a convenient tabular form and then transferred by hand into a form suitable for an ICL 1904A and the Analysis of Variance program;

3.2.1 Wait Times Experiment 1 ANOVA

Fig 5 shows the Analysis of Variance summary table as produced by the computer program. Those F ratios with a value less than 1 are not given. As can be seen some of the probabilities are exact and some directional, this is because the ANOVA program was not always sure which error variance to use therefore and omitted them.

The first and perhaps the most important point to note is that subjects are highly significantly different $p < 0.01$. This is only to be expected in such a complex task but it must be considered during further analysis.

The task however Bell or No Bell, showed no significant difference for Wait times neither did Task Order but there was a significant interaction between Task and Task Order $p = 0.064$. This is shown in graphical form in Fig 6. This indicates that while there was no significant difference due to Task Order on the Bell task there was a difference due to Task Order on the No Bell task, in the direction that when the second task was with the Bell there was only a slightly shorter response time than when the first task was with the Bell. When the No Bell task was done second however it had significantly shorter response times than when it was done first. With the No Bell task done second the response time was at its shortest of all 4 times.

Source of Variation	Sum Squares	DF	Mean Square	F	Prob
S = Subjects	1763.6	7	251.9	9.83	<0.01
A = Task order	112.1	1	112.1		
R = Subjects within groups	1651.5	6	275.2		
B = Task	3.1	1	3.1		
BS = BXS	443.9	7	63.4	1.5	
BA = BXA	203.2	1	203.2	5.06	0.064
R = BX Subjects within groups	240.7	6	40.1		
C = Commands	3457.9	6	576.3	6.1	<0.01
CS = CXS	3549.0	42	84.5		
CA = CXA	149.2	6	24.9		
R = CX Subjects within groups	3399.8	36	94.4		
D = Periods	126.4	5	25.3	1.111	
DS = DXS	727.2	35	20.8		
DA = DXA	45.1	5	9.0		
R = DX Subjects within groups	682.1	30	22.7		
BC = BXC	514.0	6	85.7	2.36	<0.05
BC S = BXCXS	1598.6	42	38.1	1.05	
BCA = BXCXA	296.5	6	49.4	1.36	
R = BCX Subjects within groups	1302.1	36	36.2		
BD = BXD	40.9	5	8.2		
BD S BXDXS	1040.2	35	29.7	1.2	
BDA = BXDXA	301.6	5	60.3	2.45	0.056
R = BDX Subjects within groups	738.6	30	24.6		
CD = CXD	692.7	30	23.1		
CD S = CXDXS	5131.4	210	24.4		
CDA = CXDXA	504.8	30	16.8		
R = CDX Subjects within groups	4626.6	180	25.7		
BC D = BXCXD	723.5	30	24.1		
BC DS = BXCXDXS	5129.0	210	24.4		
BCDA = BXCXDXA	516.7	30	17.2		
R = BCDX Subjects within groups	4612.3	180	25.6		
Total	24941.6	670			

Analysis of Variance for Wait Times Experiment 1

Fig 5

ANOVA: WAIT TIMES EXPERIMENT I

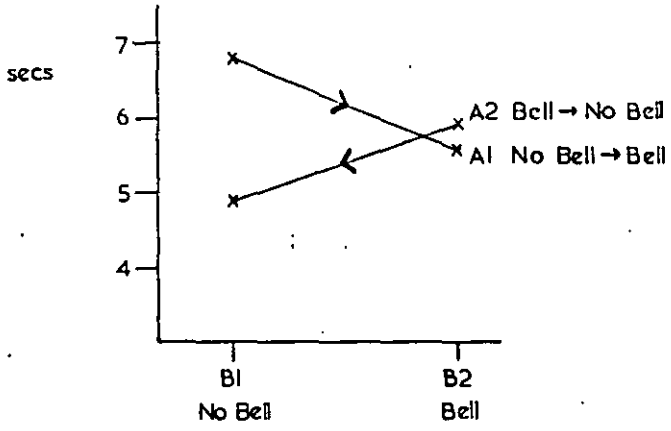


Fig 6

B x A INTERACTION $p = 0.064$

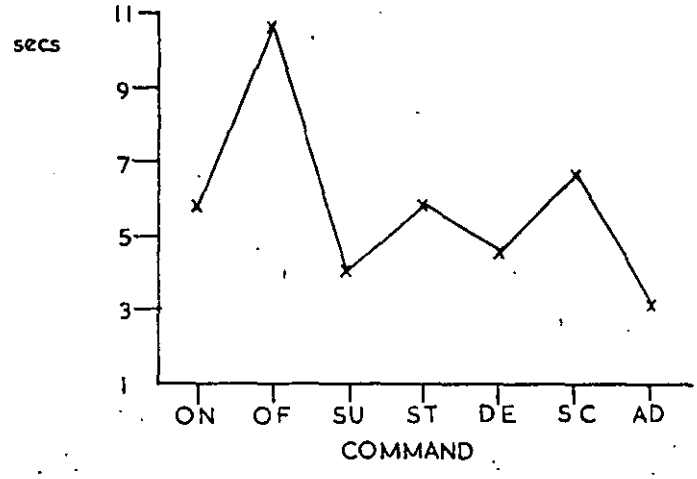


Fig 7

COMMAND EFFECT $p < 0.01$

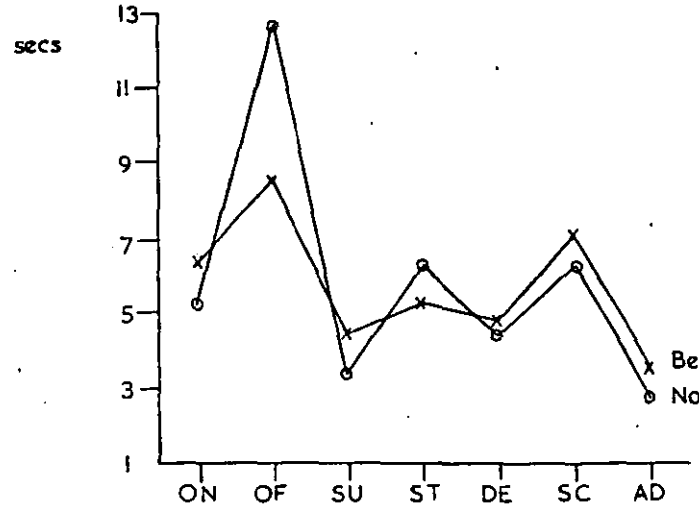


Fig 8

B x C INTERACTION $p < 0.01$

There was also a significant main effect due to Commands and this is shown in Fig 7. The points in this figure, and in succeeding similar figures, have been joined as a graph, when the X axis is not metric scale data, but I believe it gives a better indication of differences than histograms especially when 2 graphs are drawn on the same axis. With regard to Fig 7 the main point worthy of note is the high value of the Wait time before the command OFF was given and the similarity of value of all the other commands. Applying a post hoc comparison using the method due to SCHEFFE (1959) and HAYES (1970), it can be shown that the difference between the pairs of OFF-SUPPLY, OFF-DEMAND and OFF-ADD are the main contributors to the overall Command effect using $p < 0.05$. I would suggest that the large time value of OFF is due to the tactics subjects adopted in solving the problem, whereby the transformation is worked out in detail and the commands OFF then ON follow in rapid sequence when an action is selected.

The fourth significant interaction is that between Task and Commands used as shown in Fig 8 which bears, as one would expect, a remarkable resemblance to the previous figure. It shows that generally the No Bell times are shorter than the Bell times for most commands. These differences for ON, SU, DE, SC and AD are not significant but indicate that for these commands the subject was working faster. The STATUS command however is different, possibly because as this is generally a little used command the realisation of the need for STATUS output took a little longer when the subjects were under pressure with No Bell feedback The main effect is that the times for the command OFF were significantly different $p < 0.02$ in the Bell task compared with the No Bell task. I believe that the reason the OFF wait times are lower with Bell feedback is due to the fact that subjects waited for the Bell and during this

Wait time were able to consolidate their position and thoughts before detecting the change and selecting an action. The fifth significant interaction in the ANOVA summary table is the BDA interaction. This is an interaction between Task, Task Order and Period. It is almost impossible to draw and totally impossible to interpret.

3.2.2 Input times Experiment 1 ANOVA

Fig 9 shows the analysis of variance summary table as produced by the computer program. As with the Wait times the first and most important significant effect is that due to subjects. The second significant effect is an interaction between Task and Task Order similar to the Wait time. In this case however it is the Bell task that is different. Fig 10 shows this interaction graphically and indicates that irrespective of Task Order there is no significant difference in the Input times for the No Bell task. When the Bell task is done there is a significant decrease in Input time when the Bell task is done second compared with when it is done first. It should also be noted that when the Bell task is done first there is a higher Input time than either of the No Bell tasks but when the Bell task is done second there is a lower Input time than either of the No Bell tasks.

The third significant effect is that due to Commands, $p < 0.01$; this is shown graphically in Fig 11. A post hoc comparison shows that the ON, OFF and ADD commands are significantly different in their Input times to the STATUS, SUPPLY, DEMAND and SCORE commands. The reason for this is that the ON, OFF and ADD commands cannot be used effectively without parameters and that the addition of these parameters will increase the Input time. It is interesting to note that the only other command that was parameterised at all was the DEMAND command and that this is

Source of Variation	Sum Squares	DF	Mean Square	F	Prob
S = Subjects	470.9	7	67.3	16.5	<0.01
A = Task order	38.4	1	38.4		
R = Subjects within groups	432.6	6	72.1		
B = Task	5.7	1	5.7	4.03	
BS = BXS	34.6	7	4.9	3.5	
BA = BXA	26.0	1	26.0	18.04	0.006
R = BX Subjects within groups	8.6	6	1.4		
C = Commands	516.9	6	86.1	4.06	<0.01
CS = CXS	889.5	42	21.2		
CA = CXA	125.7	6	20.9		
R = CX Subjects within groups	763.8	36	21.2		
D = Periods	214.1	5	42.8	8.96	<0.01
DS = DXS	170.5	35	4.9		
DA = DXA	27.2	5	5.4	1.14	
R = DX Subjects within groups	143.3	30	4.8		
BC = BXC	23.4	6	3.9		
BC S = BXCXS	170.9	42	4.1		
BCA = BXCXA	51.9	6	8.6	2.61	0.033
R = BCX Subjects within groups	119.1	36	3.3		
BD = BXD	13.4	5	2.7		
BD S = BXDXS	193.8	35	5.5	1.414	
BDA = BXDXA	76.4	5	15.3	3.9	0.008
R = BDX Subjects within groups	117.4	30	3.9		
CD = CXD	250.4	30	8.3	2.3	<0.01
CD S = CXDXS	689.1	210	3.3		
CDA = CXDXA	48.4	30	1.6		
R = CDX Subjects within groups	640.8	180	3.6		
BC D = BXCXD	99.8	80	3.3		
BC DS = BXCXDXS	929.2	210	4.4		
BCDA = BXCXDXA	197.1	30	6.6	1.03	
R = BCDX Subjects within groups	732.1	180	4.0		
Total	4672.6	671			

Analysis of Variance for Input Times Experiment 1

Fig 9

ANOVA: INPUT TIMES EXPERIMENT I

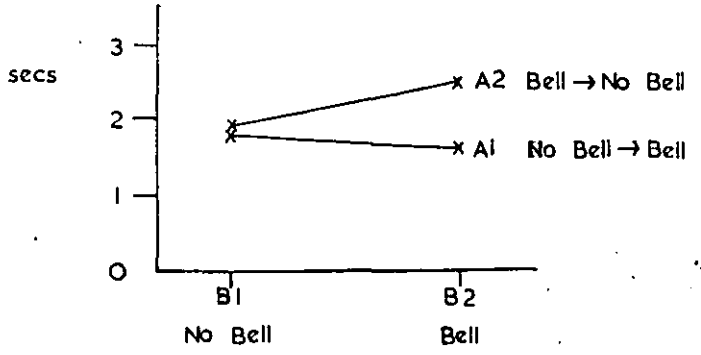


Fig 10

B x A INTERACTION p= 0·006

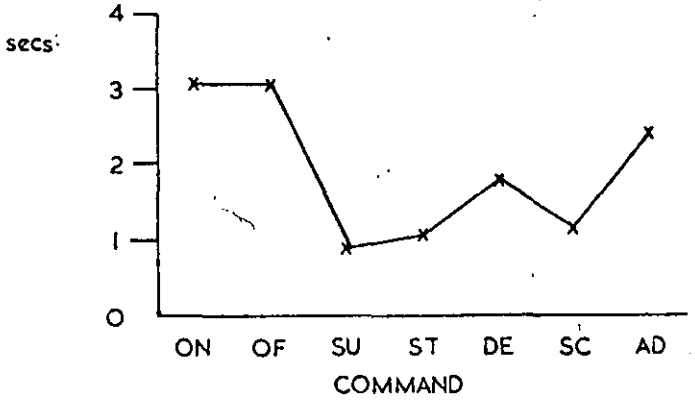


Fig 11

COMMAND EFFECT $p < 0{\cdot}01$

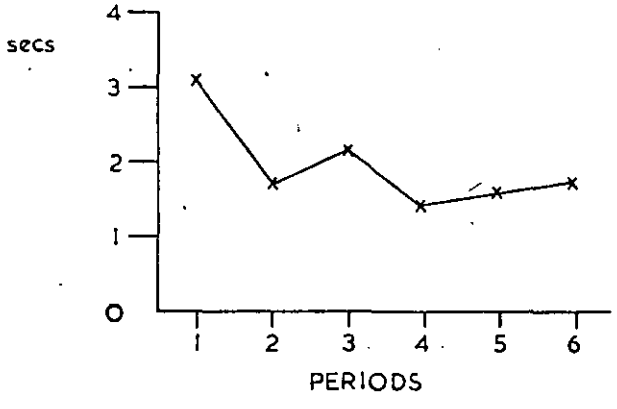


Fig 12

PERIOD EFFECT $p < 0{\cdot}01$

slightly higher than the other non-parameterised commands.

The fourth significant effect is that due to Periods, $p < 0.01$. This is shown graphically in Fig 12, which shows that over the 6-5 minute periods there is a decrease in the time taken to input Commands. There are 3 other significant interactions that are difficult to explain, these are: a Task-Command-Task Order Interaction $p < 0.03$, a Task-Period-Task Order interaction $p = 0.008$ and a Command-Period interation, $p < 0.01$. Considerable further analysis will be necessary to understand these effects.

3.2.3 Wait times + Input times for Experiment 1 ANOVA

Fig 13 shows the summary table produced by the computer program for the sum of the Wait times and Input times previously analysed separately. Subjects have highly significant effect, $p < 0.01$. The second significant interaction is the Task-Task Order interaction shown graphically in Fig 14 and is in fact an almost exact combination of the Input time and Wait time graphs. Showing both the effect of decrease in Input time due to the Bell task and the difference in the No Bell Wait time due to Task Order.

A third significant interaction is the Command effect $p < 0.01$ shown graphically in Fig 15. This is similar to the Wait time graph Fig 7 being reduced slightly by the similarity between ON and OFF commands due to the Input time effect. Perhaps the most striking effect is that due to Periods Fig 16 with $p < 0.01$. This shows an overall decrease in Input and Wait time of some 22% over the trial. There is one further interaction Task-Periods-Task Order $p = 0.011$ which is uninterpretable. It is notable that the factor of subjects has a highly

Source of Variation	Sum Squares	DF	Mean Square	F	Prob
S = Subjects	2137.0	7	305.3	8.96	<0.01
A = Task order	19.3	1	19.3		
R = Subjects within groups	2117.7	6	352.9		
B = Task	.4	1	.4		
BS = BXS	667.3	7	95.3		
BA = BXA	374.6	1	374.6	7.67	0.032
R = BX Subjects within groups	292.7	6	48.8		
C = Commands	5077.2	6	846.2	5.417	<0.01
CS = CXS	6130.5	42	145.9		
CA = CXA	507.6	6	84.6		
R = CX Subjects within groups	5622.9	36	156.2		
D = Periods	380.5	5	76.1	3.38	<0.05
DS = DXS	606.7	35	17.3		
DA = DXA	11.3	5	2.3		
R = DX Subjects within groups	595.4	30	19.8		
BC = BXC	488.2	6	81.4	1.67	
BC S = BXCXS	2124.1	42	50.6		
BCA = BXCXA	370.5	6	61.7	1.26	
R = BCX Subjects within groups	1753.6	36	48.7		
BD = BXD	42.4	5	8.5		
BD = BXDXS	1331.2	35	38.0		
BDA = BXDXA	502.6	5	100.5	3.64	0.011
R = BDX Subjects within groups	828.6	30	27.6		
CD = CXD	1124.4	30	37.5		
CD S = CXDXS	6429.6	210	30.6		
CDA = CXDXA	523.5	30	17.4		
R = CDX Subjects within groups	5906.1	180	32.8		
BC D = BXCXD	850.0	30	28.3		
BC DS = BXCXDXS	6737.9	210	32.1		
BCDA = BXCXDXA	611.6	30	20.4		
R = BCDX Subjects within groups	6126.3	180	34.0		
Total	34127.6	671			

Analysis of Variance for Wait and Input Times Experiment 1

Fig 13

ANOVA: WAIT & INPUT TIMES EXPERIMENT I

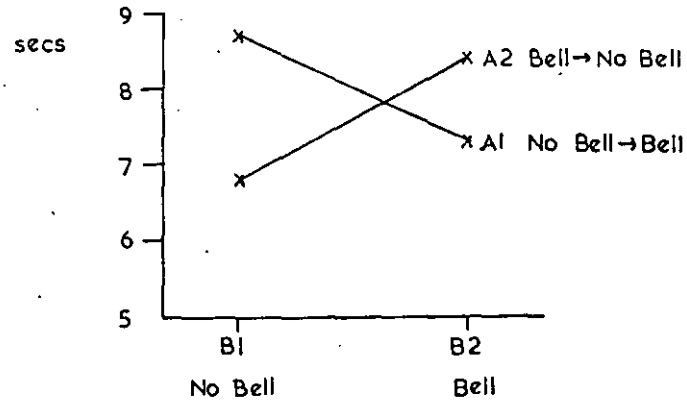


Fig 14

B x A INTERACTION $p = 0.032$

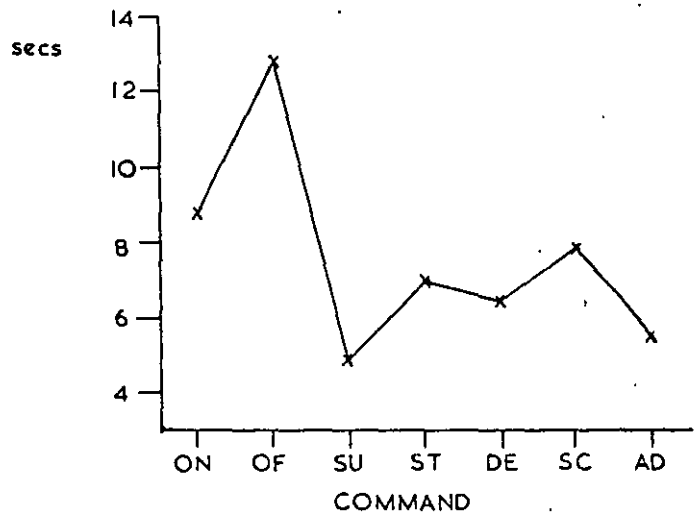


Fig 15

COMMAND EFFECT $p < 0.01$

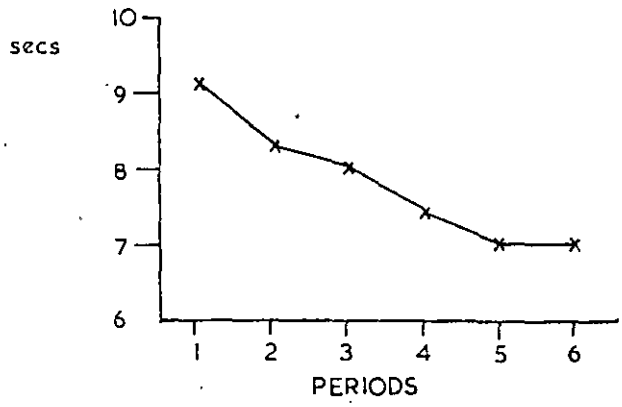


Fig 16

PERIOD EFFECT $p < 0.01$

significant effect but that none of the interaction effects that are significant include subjects as a factor. This means that conclusions can be drawn regarding other effects without worrying about individual differences.

3.2.4 Wait Times Experiment 2 ANOVA

Fig 17 shows the analysis of variance summary table for the Wait times for Experiment 2. Again subjects have a highly significant effect on Wait times. A second significant effect is that due to Task, $p < 0.01$: this is shown graphically in Fig 18 and shows that the TIC task Wait time is significantly lower than the SIC task Wait Times. Indicating that subjects thought more and therefore waited longer before command entry when they were in control of the changes.

The third significant effect $p < 0.01$ is shown graphically in Fig 19 which shows striking differences between command Wait times. OFF and ADD having the longest Wait times. This suggests that subjects did most of their thinking before selecting OFF or ADD.

The 2 other interactions can best be considered together. They are a Command-Task Order interaction, $p = 0.013$ as shown in Fig 20 and a Command-Task interaction, $p < 0.05$ shown in Fig 21. These 2 show how the previous effect due to Commands alone is made up. Taking the Command-Task interaction first, this shows difference in command OFF[(2)] time, it taking considerably longer to issue an OFF command in the SIC task. Also highly significant is the ADD[(7)] Wait time which is significantly longer in the SIC task than in the TIC task. It is also interesting to note that the SCORE command[(6)] had a longer Wait time in the SIC task than in the TIC task. This is presumably because the SCORE command

Source of Variation	Sum Squares	DF	Mean Square	F	Prob
S = Subjects	2460.1	7	351.4	8.02	<0.01
A = Task order	151.6	1	151.6		
R = Subjects within groups	2308.5	6	384.7		
B = Task	1421.8	1	1421.8	5.09	<0.01
BS = BXS	1722.6	7	246.1		
BA = BXA	46.7	1	46.7		
R = BX Subjects within groups	1675.9	6	279.3		
C = Commands	10209.7	6	1701.6	10.31	<0.01
CS = CXS	9101.0	42	216.7		
CA = CXA	3160.9	6	526.8	3.19	0.013
R = CX Subjects within groups	5940.1	36	165.0		
D = Periods	103.5	5	20.7		
DS = DXS	1946.8	35	55.6		
DA = DXA	486.9	5	97.4	2.00	
R = DX Subjects within groups	1459.9	30	48.6		
BC = BXC	3548.2	6	591.4	3.00	<0.05
BC S = BXCXS	7620.8	42	181.4		
BCA = BXCXA	583.2	6	97.2		
R = BCX Subjects within groups	7037.6	36	195.5		
BD = BXD	178.9	5	35.9		
BD S = BXDXS	1526.5	32	47.7		
BDA = BXDXA	657.5	5	131.5	4.08	0.007
R = BDX Subjects within groups	868.9	27	32.2		
CD = CXD	2161.8	30	72.1		
CD S = CXDXS	11585.6	210	55.2		
CDA = CXDXA	3206.1	30	106.9	2.29	0.001
R = CDX Subjects within groups	8379.6	180	46.6		
BC D = BXCXD	1842.8	30	61.4		
BC DS = BXCXDXS	9069.4	192	47.2		
BCDA = BXCXDXA	1978.9	30	66.0		
R = BCDX Subjects within groups	7090.5	162	43.8		
Total	64499.6	650			

Analysis of Variance for Wait Times Experiment 2

Fig 17

ANOVA: WAIT TIMES EXPERIMENT 2

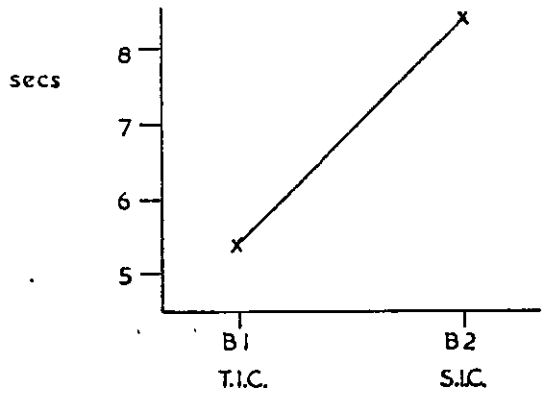


Fig 18

TASK EFFECT $p < 0.01$

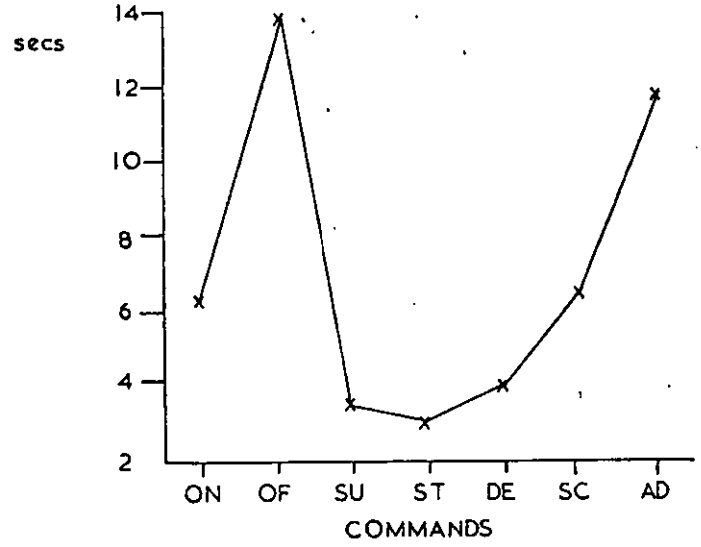


Fig 19

COMMAND EFFECT $p < 0.01$

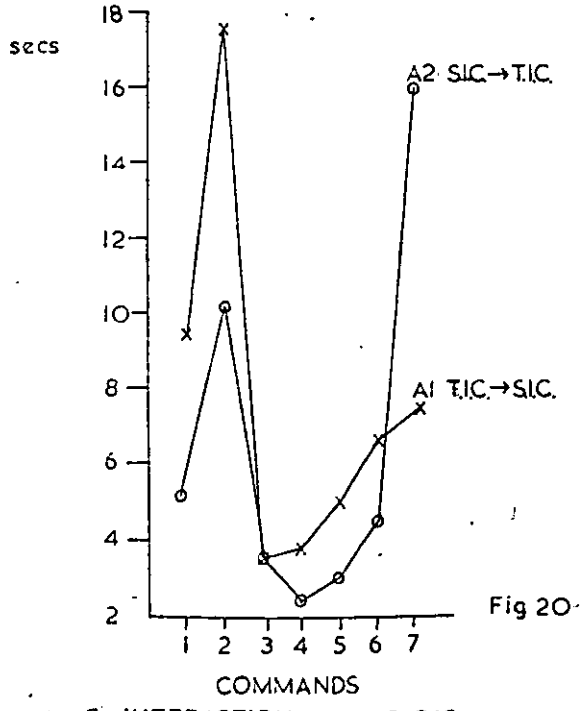


Fig 20

A x C INTERACTION $p = 0.013$

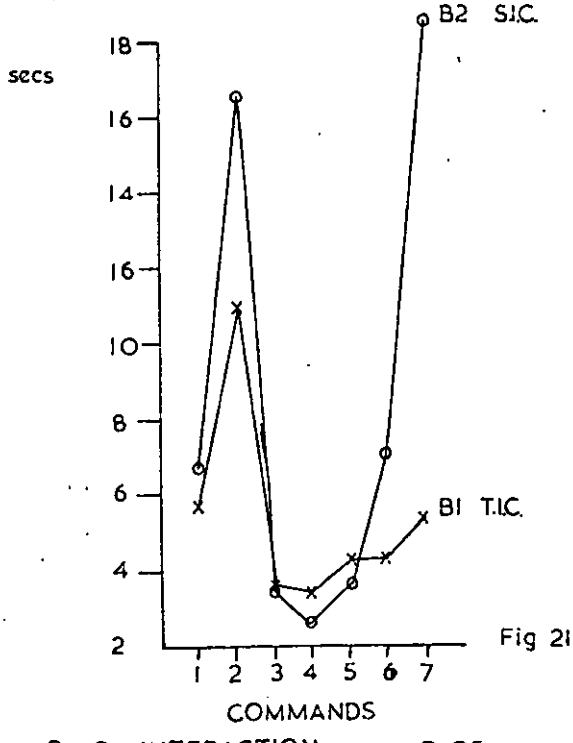


Fig 21

B x C INTERACTION $p < 0.05$

was responsible for initiating a change in the SIC task and that therefore more thought and hence more time was needed before it was issued. The Task Order-Command interaction Fig 21 has the same effect for the OFF command. It is also apparent that when the SIC task was done before the TIC task more time was spent on the ADD commands than when the TIC task was done before the SIC task. This indicates that when the subjects had done the timebase initiated task first they felt less need to spend time on the ADD command than when they did the SIC task first.

3.2.5 Input times Experiment 2 ANOVA

Fig 22 shows the analysis of variance summary table for the Input times for Experiment 2. Subjects again are a highly significant effect. The second significant effect is a Command effect. This is shown graphically in Fig 23. A post hoc comparison shows the On, OFF and ADD, are contributing significantly to the overall effect. This is due to the same effect as noticed in Experiment 1, that these are the commands that have to be parameterized and therefore take longer to type. The third significant interaction is a period effect $p < 0.01$. This is shown graphically in Fig 24, and is very similar in appearance to Fig 12 for Experiment 1. It shows a decreasing input typing time over the trial.

The fourth significant interaction is a Task-Command interaction and is shown graphically in Fig 25. This again shows the difference in the use of the ADD Command, much longer time being spent on the ADD command in the SIC task than in the TIC task. It is also possible to suggest that the nearly identical values of ON and OFF Input times in the TIC task as compared with the different Input times for the SIC tasks, were caused

Source of Variation	Sum Squares	DF	Mean Square	F	Prob
S = Subjects	497.3	7	71.0	13.11	<0.01
A = Task order	13.2	1	13.2		
R = Subjects within groups	484.2	6	80.7		
B = Task	64.4	1	64.4	3.77	
BS = BXS	179.1	7	25.6		
BA = BXA	76.7	1	76.7	1.97	0.077
R = BX Subjects within groups	102.4	6	17.1		
C = Commands	1446.1	6	241.0	16.56	<0.01
CS = CXS	643.0	42	15.3		
CA = CXA	119.1	6	19.8	1.364	
R = CX Subjects within groups	523.9	36	14.6		
D = Periods	186.5	5	37.3	4.913	<0.01
DS = DXS	298.9	35	8.5		
DA = DXA	71.2	5	14.2	1.87	
R = DX Subjects within groups	227.7	30	7.6		
BC = BXC	489.8	6	81.6	6.31	<0.01
BC S = BXCXS	503.9	42	12.0		
BCA = BXCXA	38.3	6	6.4		
R = BCX Subjects within groups	465.6	36	12.9		
BD = BXD	99.9	5	19.9	2.27	
BD S = BXDXS	317.0	32	9.9		
BDA = BXDXA	79.5	5	15.9	1.8	
R = BDX Subjects within groups	237.5	27	8.8		
CD = CXD	304.7	30	10.2		
CD S CXDXS	1802.9	210	8.6		
CDA = CXDXA	397.2	30	13.2	1.69	0.019
R = CDX Subjects within groups	1405.7	180	7.8		
BC D = BXCXD	331.5	30	11.0		
BC DS = BXCXDXS	1108.9	192	5.8		
BCDA = BXCXDXA	231.2	30	7.7		
R = BCDX Subjects within groups	877.6	162	5.4		
Total	8273.9	640			

Analysis of Variance for Input times Experiment 2

Fig 22

ANOVA: INPUT TIMES EXPERIMENT 2

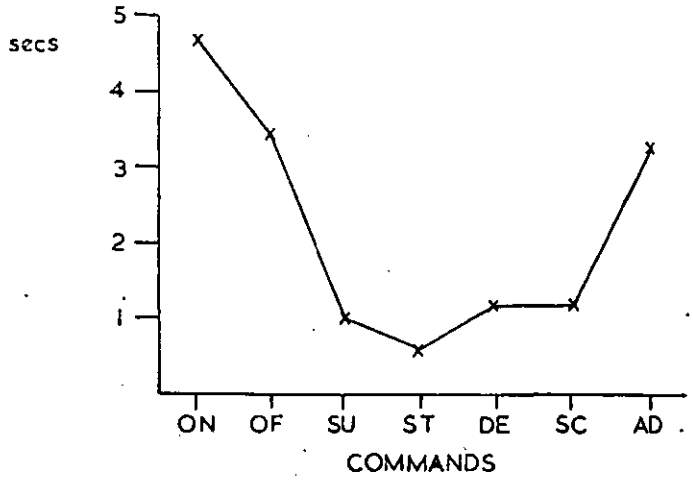


Fig 23

COMMAND EFFECT $p < 0{\cdot}01$

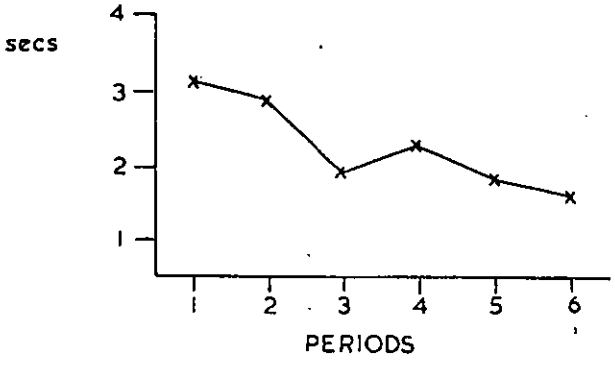


Fig 24

PERIOD EFFECT $p < 0{\cdot}01$

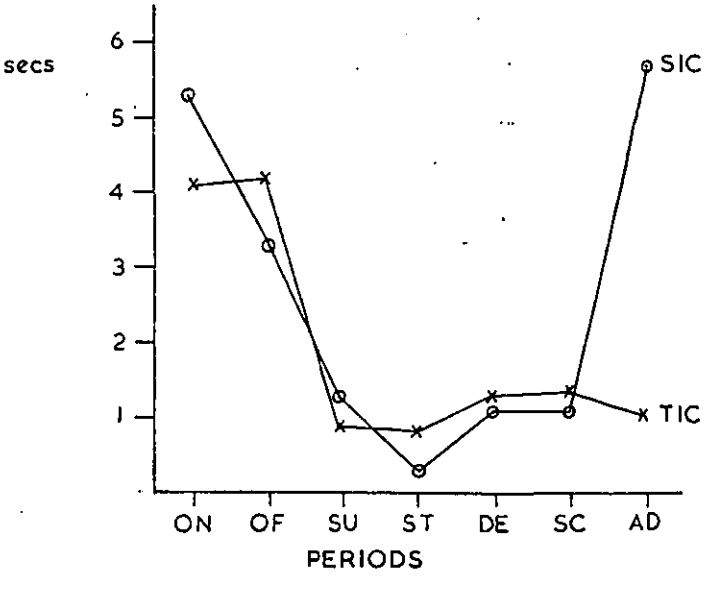


Fig 25

B x C INTERACTION: $p = 0{\cdot}011$

by a difference in strategy. Some subjects in the SIC task adopted the strategy of turning all the areas OFF then re-selecting ON those areas required. This produces a shorter Input time for the command OFF and a longer Input time for the command ON. This strategy is not acceptable in the TIC task because of the continuous evaluation process.

3.2.6 Wait and Input Times Experiment 2 ANOVA

Fig 26 shows the analysis of variance summary table for the addition of the Wait and Input times Experiment 2. Again subjects are a highly significant factor. All the effects shown in this summary table and depicted graphically in Figs 27-30 are due entirely to the observed Wait time effects and have no significant part due to the combination of Wait and Input times.

3.2.7 Summary of Analysis of Variance Data

Taking Experiment 1 first, it can be shown, with the Input time data, that there are differences in performance between the Bell and No Bell task. Perhaps the most significant result is the Task-Task Order interaction of Fig 14 showing a combined interaction due to Input time and Wait time, where the Input typing time for the Bell task is significantly different due to Task Order and the Wait time for the No Bell is significantly different due to Task Order. The effects noted due to Command and Command-Task interaction are predictable and expected and show features due to the Task under consideration. The Period effects are just those to be expected in these types of, no pre-learning, experiments. In Experiment 2 the analysis of variance techniques have shown a striking difference between the 2 tasks in terms of Wait time and some interesting effects of Command-Task and Command-Task Order interactions, indicating different strategies are used for

Source of Variation	Sum Squares	DF	Mean Square	F	Prob
S = Subjects	4045.3	7	577.9	10.7	<0.01
A = Task order	75.4	1	75.4		
R = Subjects within groups	3969.8	6	661.6		
B = Task	2091.6	1	2091.6	5.12	<0.01
BS = BXS	2503.7	7	357.7		
BA = BXA	55.2	1	55.2		
R = BX Subjects within groups	2448.4	6	408.1		
C = Commands	17001.0	6	2833.5	12.8	<0.01
CS = CXS	12055.9	42	287.0		
CA = CXA	4134.7	6	689.1	3.13	0.014
R = CX Subjects within groups	7921.1	36	220.0		
D = Periods	450.7	5	90.1		
DS = DXS	2841.7	35	81.2		
DA = DXA	824.9	5	164.9	2.45	0.055
R = DX Subjects within groups	2016.7	30	67.2		
BC = BXC	6034.5	6	1005.7	4.0	<0.01
BC S = BXCXS	9624.5	42	229.1		
BCA = BXCXA	600.0	6	100.0		
R = BCX Subjects within groups	9024.5	36	250.7		
BD = BXD	336.8	5	67.4		
BD S = BXDXS	2455.2	32	76.7		
BDA = BXDXA	999.9	5	199.9	3.7	0.011
R = BDZ Subjects within groups	1455.2	27	53.9		
CD = CXD	2725.1	30	90.8	1.31	
CD S = CXDXS	16968.6	210	80.8		
CDA = CXDXA	4323.9	30	150.8	2.18	0.001
R = CDX Subjects within groups	12444.6	180	69.1		
BC D = BXCXD	2474.8	30	82.5		
BC DS = BXCXDXS	11461.3	192	59.7		
BCDA = BXCXDXA	2743.5	30	91.4		
R = BCDX Subjects within groups	8717.8	162	53.8		
Total	93070.8	650			

Analysis of Variance for Wait and Input times Experiment 2

Fig 26

ANOVA WAIT & INPUT TIMES EXPERIMENT 2

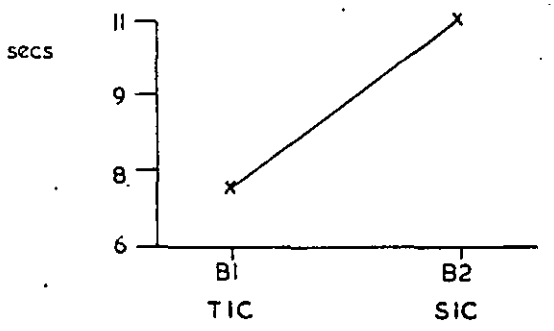


Fig 27

TASK EFFECT $p < 0.01$

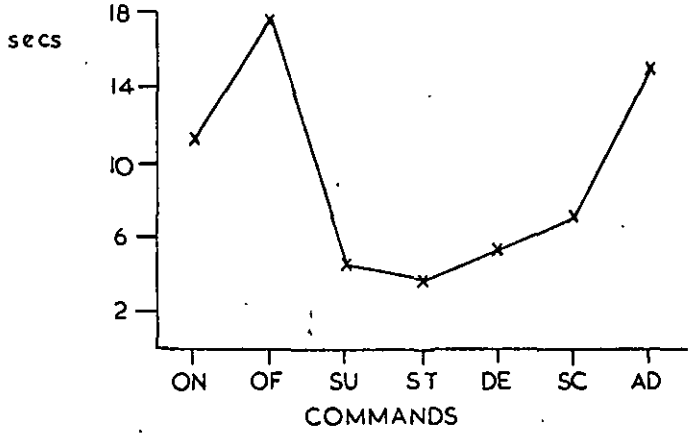


Fig 28

COMMAND EFFECT $p < 0.01$

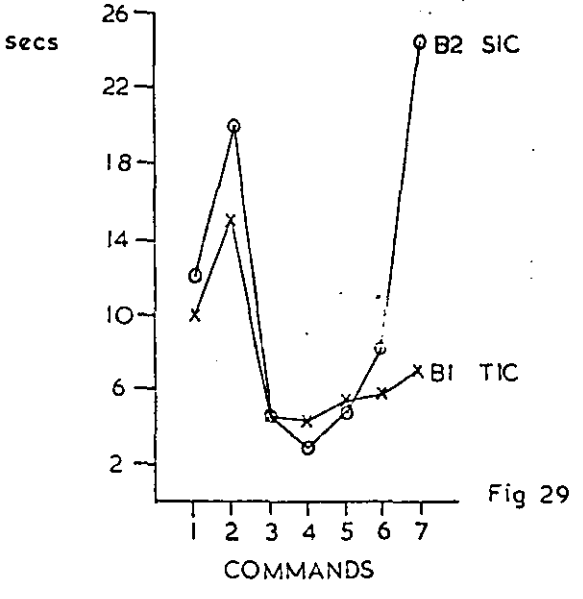


Fig 29

B x C INTERACTION $p < 0.01$

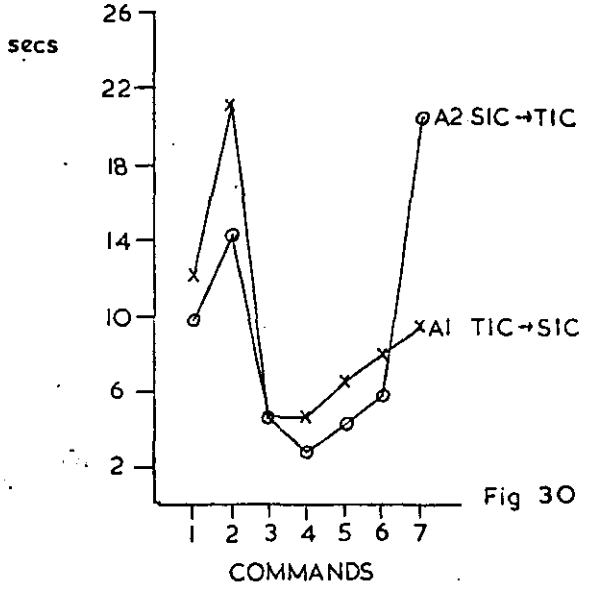


Fig 30

A x C INTERACTION $p = 0.014$

task differing only in their relationship with time.

3.3 Command Sequence Analysis

This analysis is concerned with identifying the strategic and tactical decision processes that were used by the subjects. The analysis is founded on the premise that all the information needed to analyse the decision making strategies is contained within the sequence of commands issued, as this is the only way the subject can influence the system. As every command that was issued was recorded, evidence of the decision making strategies adopted must be contained within this recorded data. A preliminary look showed that subjects issued a large spectrum of total commands and it was obvious that looking at strings of up to 250 numbers was an ineffective method of deciding whether there were repeating cyclic patterns either within or across subjects.

To present the data in visual form 4 graphs were constructed one for each group of subjects. These graphs are shown in Figs 31-34 corresponding to groups 1-4. The horizontal axis is time-based command sequence, real time being removed. The different lengths of the graphs representing the different numbers of commands used by subjects. Eight separate graphs are plotted in each Figure except for Figs 31 and 32, unfortunately the computer destroyed the data for one half of one subject in each of groups 1 and 2, after the analysis of variance tests were done. The identification on the left of each of the 8 graphs refers to subject number, group number and trial number. Within each of these small graphs the vertical axis is an arbitrary command number, the top point is ADD and the next is SCORE then DEMAND, STATUS, SUPPLY, OFF and ON down the axis. This axis is only nominal in its level of measurement. These graphs then show the picture of commands issued.

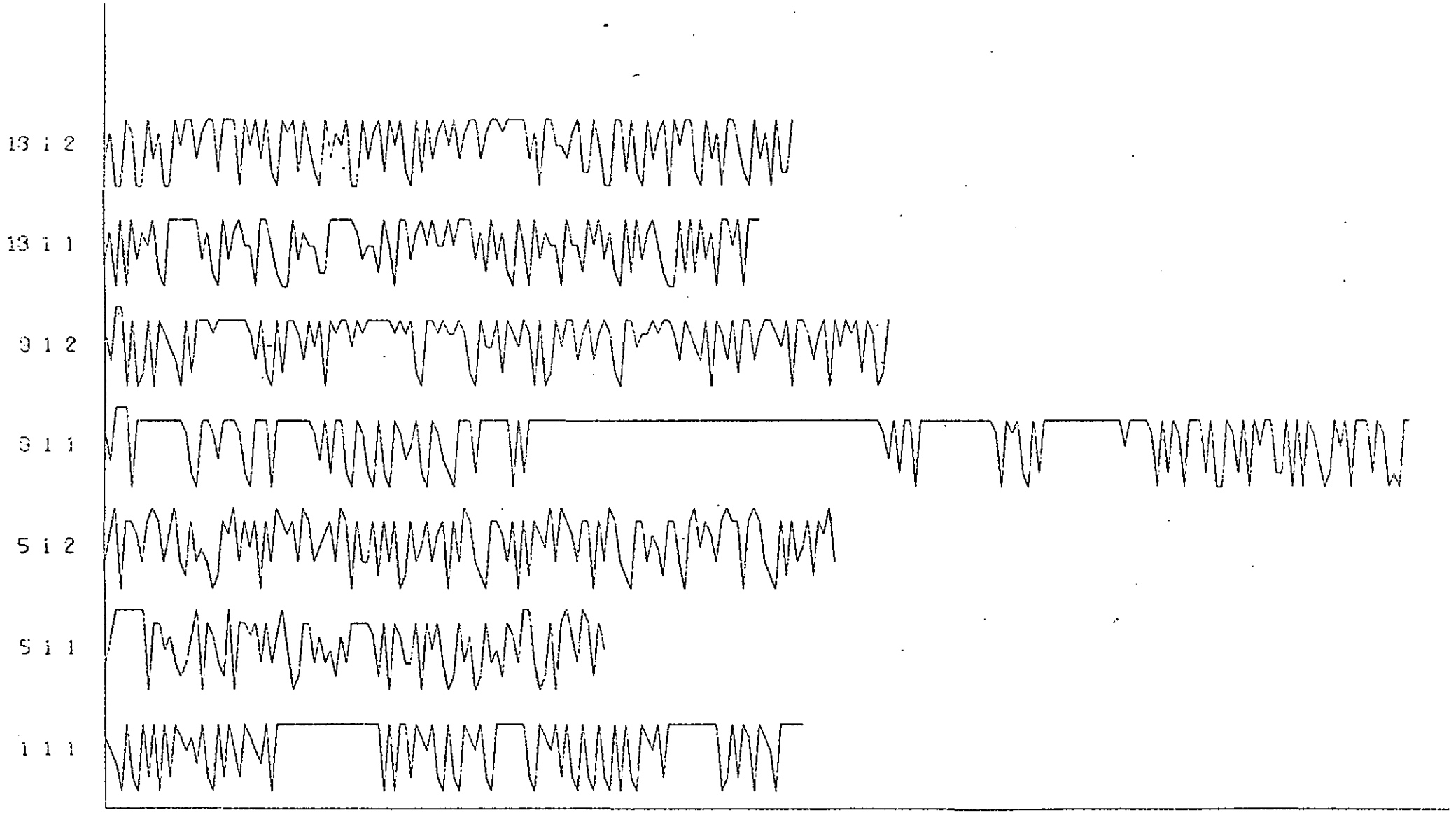


Fig 31. TIC No Bell then TIC Bell

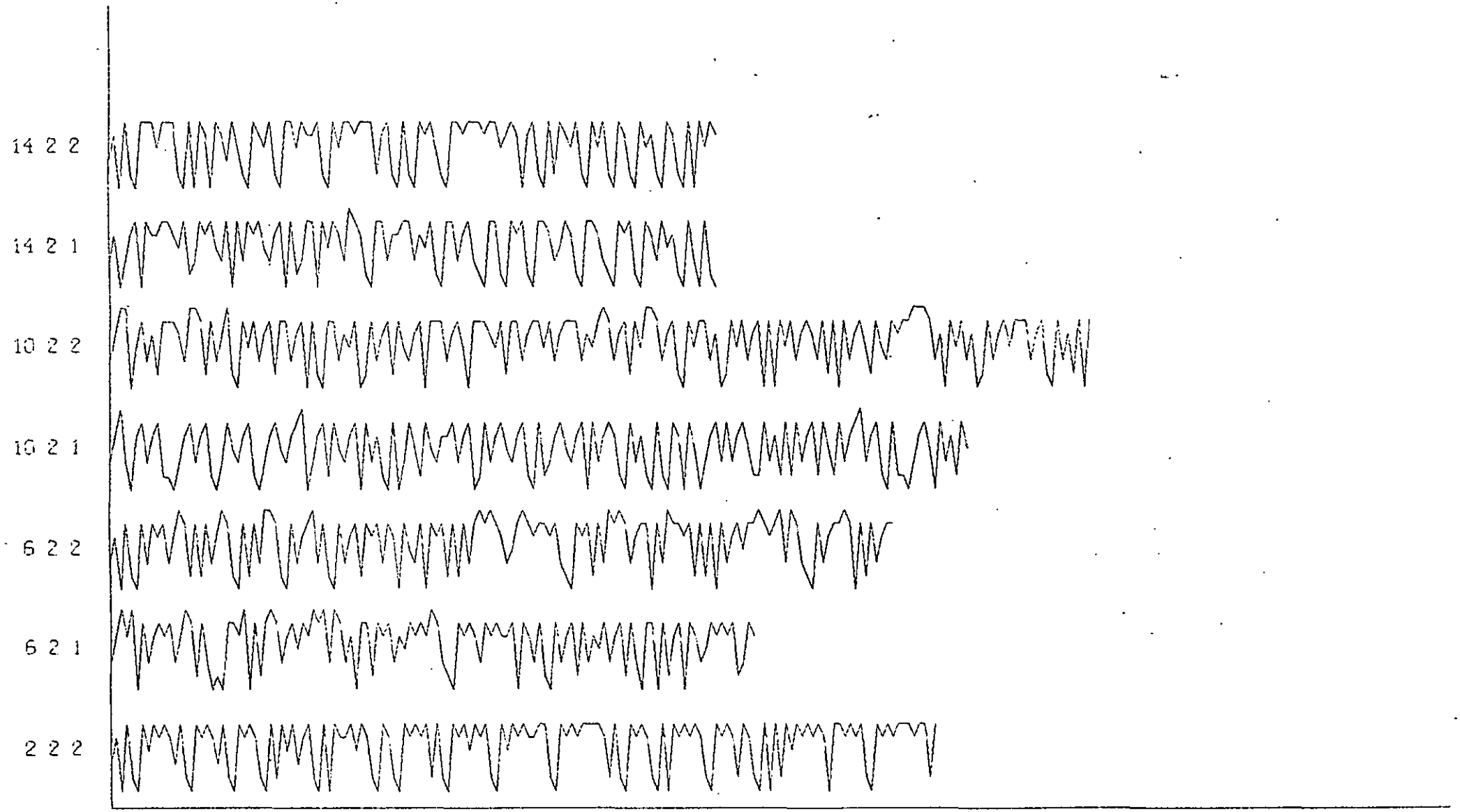


Fig 32. TIC Bell then TIC No Bell

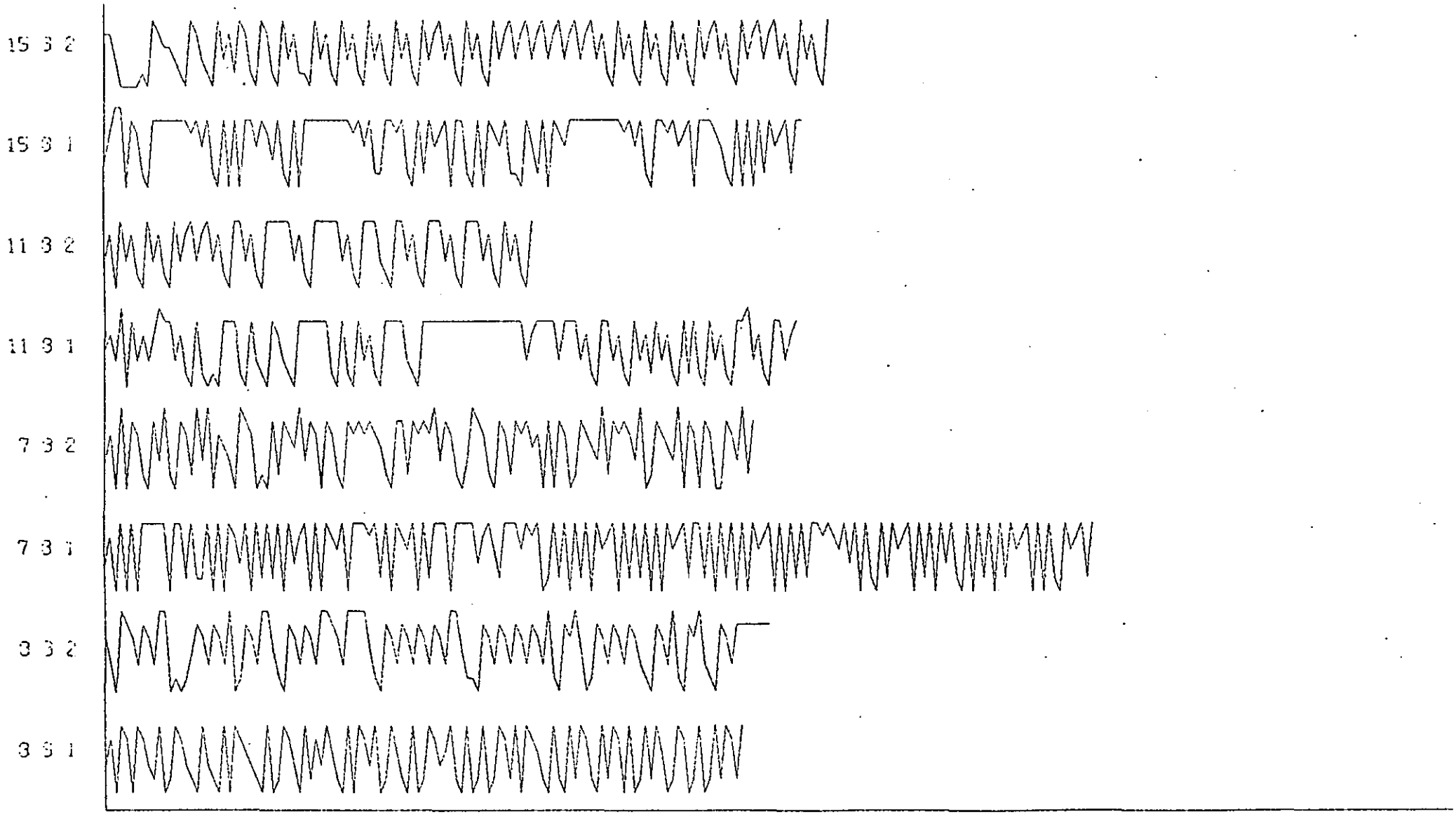
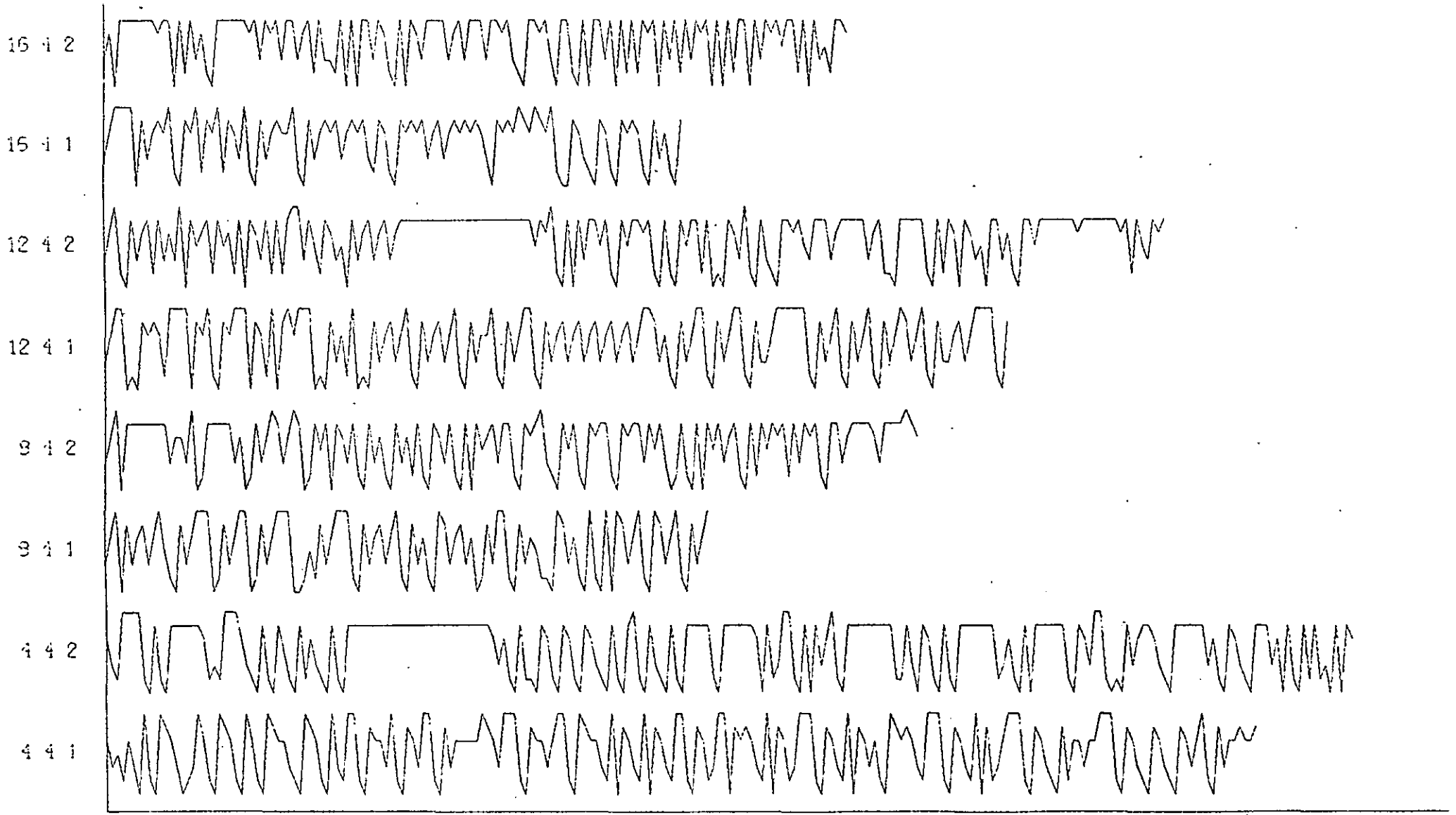


Fig 33. TIC No Bell then SIC



GROUP 4 COMMAND SEQUENCE

Fig 34. SIC then TIC No Bell

Some interesting points can be seen from these graphs. Subject 9-1-1, Fig 31 has long periods where only the command SCORE(6) was issued, this is a monitoring strategy on the No Bell TIC task. Fig 32, subject 2-2-2 an interesting strategy can be seen oscillating between the SCORE command(6) and DEMAND command(5) through the trial. In Fig 33, subject 15-3-2 has a very regular pattern of commands issued. This is in fact a subject doing the on-line SIC task and therefore as events were under his control the pattern is much more cyclic. Subject 7-3-1 on the same Figure adopted a purely oscillating strategy switching areas ON and OFF more or less at random without any particular regard for their STATUS or DEMANDS in an attempt to balance the system. Another interesting point is that all subjects on their first trial have no pattern during the first 15 commands, after this some subjects begin to settle down to a recognisable pattern.

Careful observations were made of these 4 figures and from them likely sequence patterns were determined. The patterns were fed as Input data into a computer program which counted the frequency of occurrence of the patterns across all subjects and groups. Of the 13 patterns tried only 3 showed significant differences across task groups. These were:

1. A sequence of 6 score commands, which occurred only in the No Bell TIC task, it represents a monitoring strategy used in the absence of change feedback. This is the only sequence that could differentiate between the Bell task and the No Bell task.
2. A sequence SCORE, STATUS, ON or OFF, which occurred in the TIC tasks significantly more often than the SIC tasks. I suggest this is because in the SIC task most subjects used the all OFF strategy and hence had no need for status

information.

3. A sequence SCORE, ON or OFF, SCORE, which never occurred in the SIC task. This sequence indicates action taken in a hurry without consideration of demand or supply and this never happened in the SIC tasks.

It is apparent that not all the information in these sequences has been extracted. It was hoped that a selective approach would prove feasible but it is now apparent that a more exhaustive 'number crunching' approach is necessary to maximise information from the available data. Other problems occur in this sort of analysis; one is errors made by subjects in entering the wrong command, which alters the sequences. The minor transposition of orders in command sequences having little significance from a strategic viewpoint make sequence analysis very difficult. The third problem is that, in the experiments as conducted, the subjects could look up in the paper roll produced by the teletype information about previous decisions and occurrences, so that not every piece of information they used was requested from the computer.

If a way could be found to negotiate these difficulties I am certain that command sequence analysis would yield valuable data regarding the decision making process of the subjects.

3.4 Errors

In both experiments the subjects performance at optimisation was recorded at 10 second intervals throughout the trial. This performance data is converted into an error score by an unusual technique.

The error score used in all subsequent analysis is in fact composed by

counting the number of ways optimisations could have been achieved that were better than the one actually chosen. This was done for errors of overloading and underloading. This technique has the advantage of yielding a relative score across subjects and tasks and does not depend on the numbers actually being manipulated. It is a measure of how much better the subject could have done at any point in the trial.

3.4.1 Errors against time

The graphs in Figs 35-38 are based on the same axis as the graphs of command sequence previously described. For each subject at each command, the error is plotted. The graphs show the distribution of error scores, the vertical lines across the graphs represent changes in the system caused either by the subject for SIC tasks or by the system for TIC tasks. Comparing the 2 sets of graphs it can be seen that when subjects had a continuous sequence of 6 'SCORE's' as in Fig 31, subject 9-1-1 then in the error scores Fig 35, subject 9-1-1 the error is zero. A count was made of zero crossings and error reversals but no significance could be detected across groups, although there was a significant difference between subjects $p < .01$ for both measures, SIEGEL (1959).

3.4.2 Cumulative error scores

The picture presented by the plots of errors is somewhat confused and to make the interpretation more understandable the results were plotted on a cumulative basis. Figs 1-6 in Annex C show the absolute cumulative sum (cusum) of the errors for each group on each trial at 10 second intervals. Absolute, in this context, means that the direction of the error, (overload or underload) is ignored and all errors treated alike. As can be seen from the graphs the errors cumulated steadily throughout

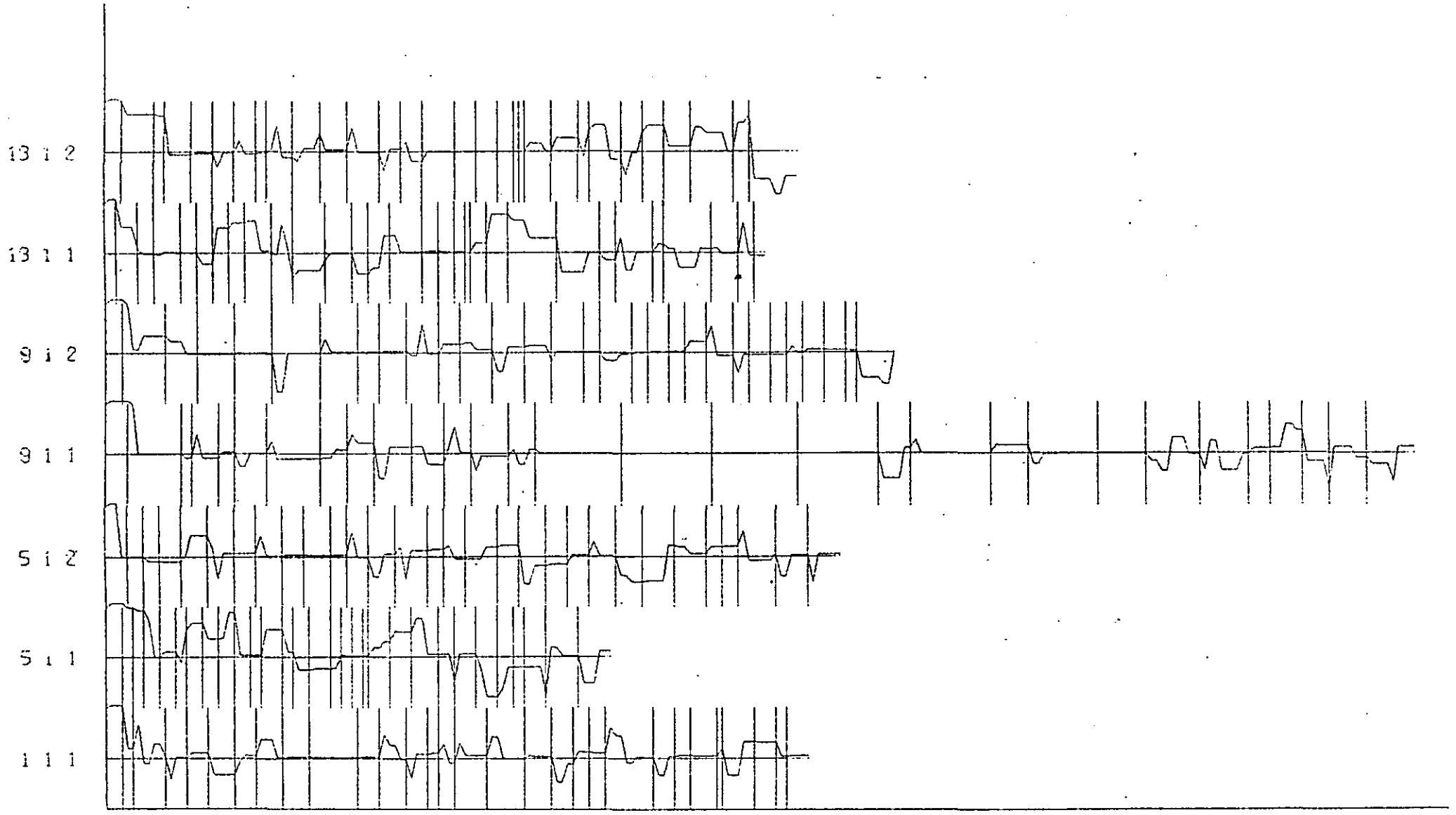


Fig 35. TIC No Bell then TIC Bell

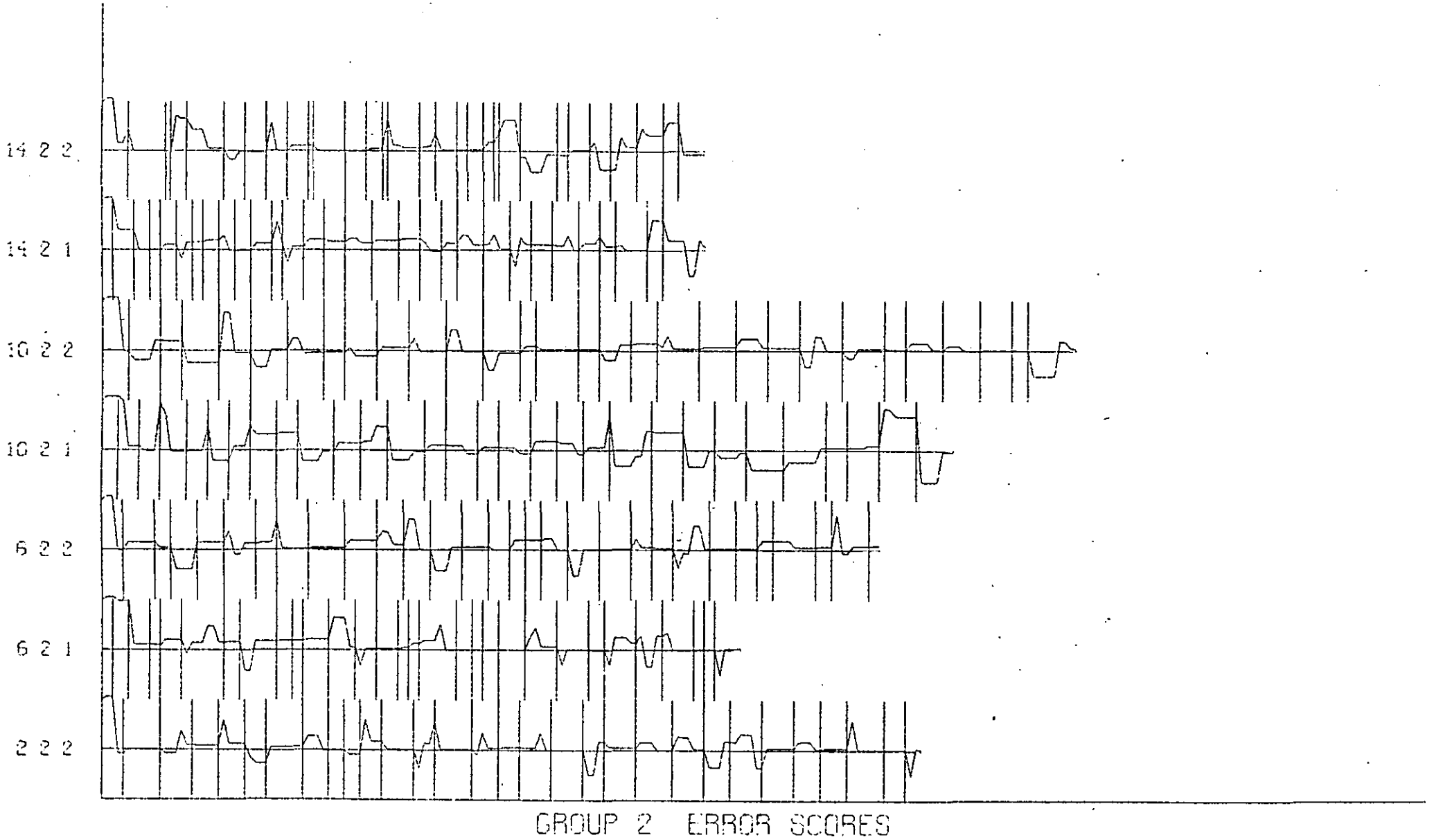


Fig 36. TIC Bell then TIC No Bell

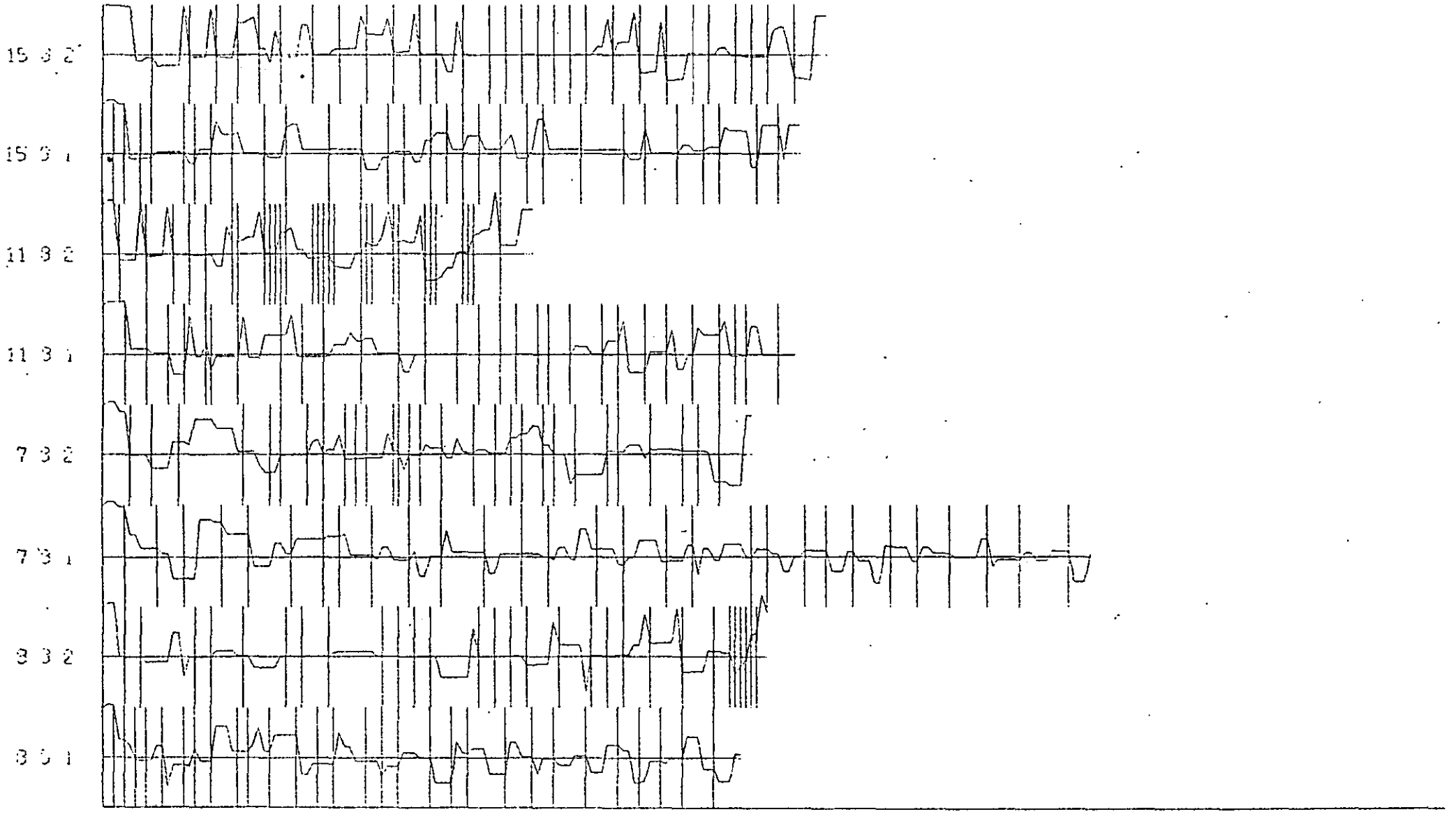


Fig 37. TIC No Bell then SIC

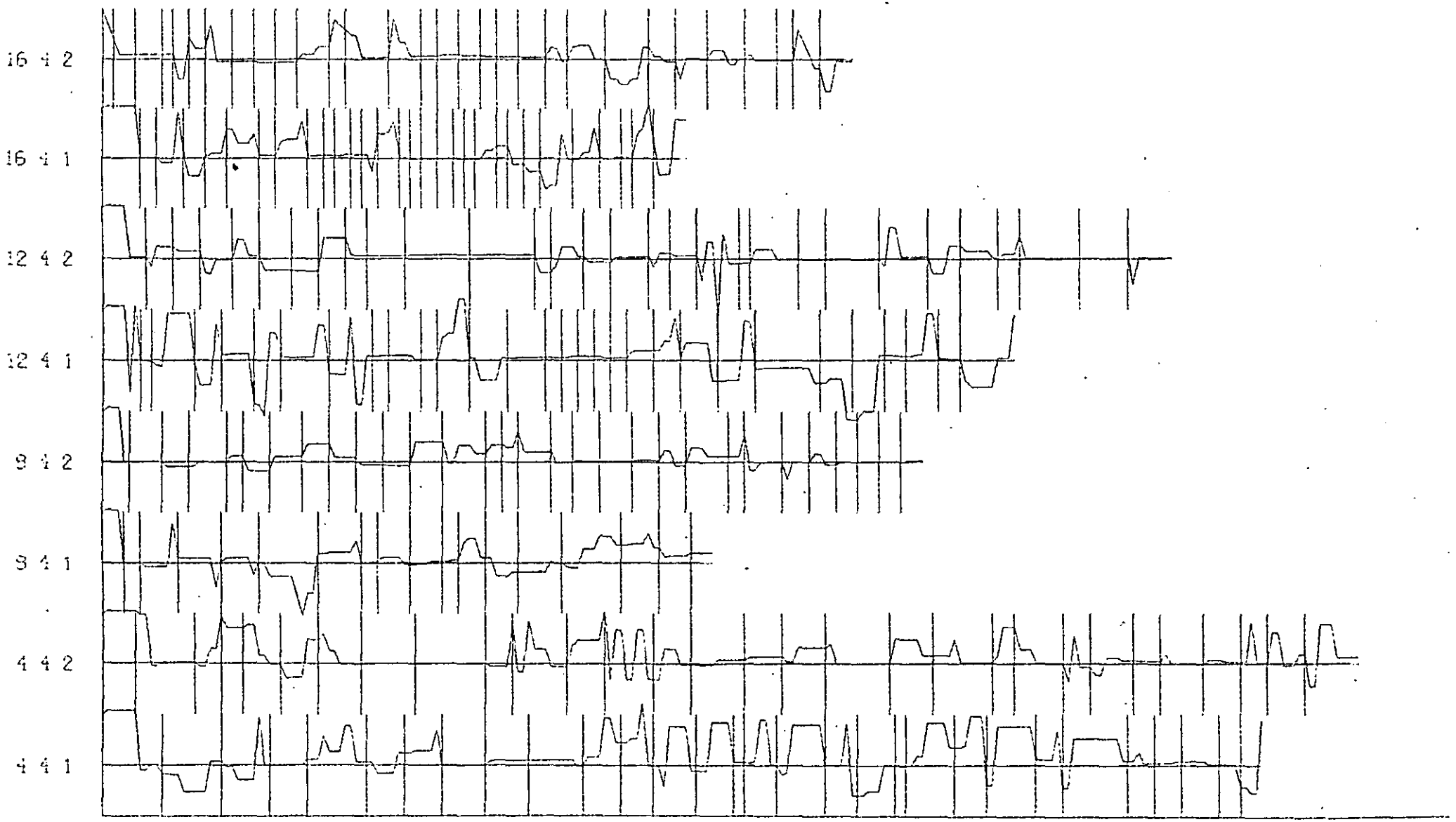


Fig 38. SIC then TIC No Bell

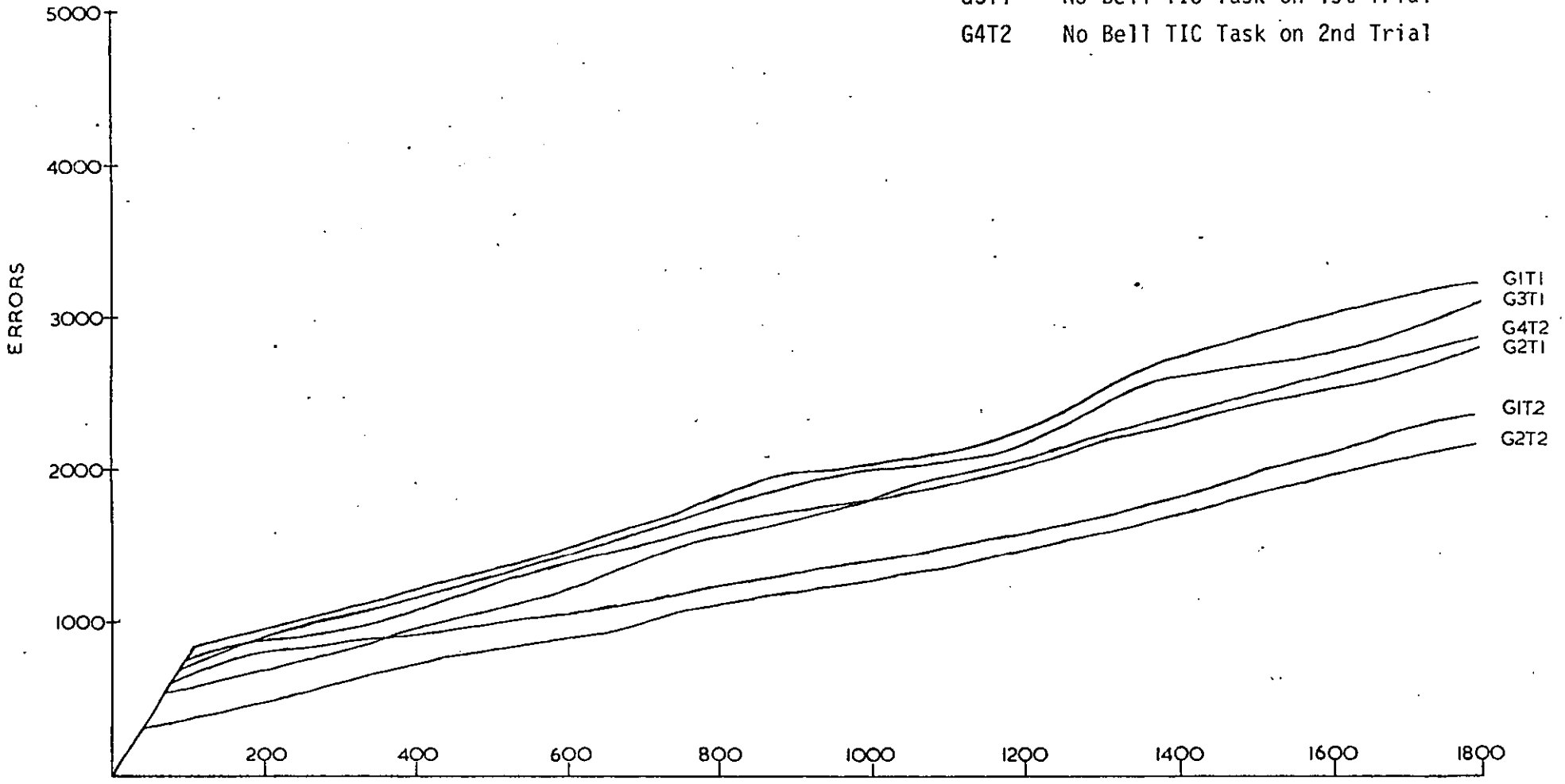
the task after an initial early rush. The main slope of the graphs represents the subjects toleration of error and can be seen to be very nearly the same for all subjects. The terminal values for each subjects error score for each group was compared using χ^2 test and no significant different across subjects, $p < 0.2$ could be determined.

To simplify the appreciation of the inter-group differences; the mean of each group was calculated and is plotted in Fig 39. This includes all the trials done with the TIC task including those done in Experiment 2.

It can be seen that the worst error performance was achieved by the 2 groups who did the TIC No Bell task first, Group 1 Trial 1 and Group 3 Trial 1. There is no significant difference between these curves, indicating a consistency of results across the 2 experiments.

Considering the second trials of Experiment 1 the best performance of all was achieved by Group 2 Trial 2. This is in fact the No Bell task on the second trial having started with the Bell task. This performance is significantly better than Group 1 Trial 2 who did the Bell task second. This indicates that the Bell assists the subjects when they first try the task and perhaps induces a false sense of ease when present for the second task, there are other alternative explanations and further research is needed to understand these effects. The slopes on the graphs for the second trials Experiment 1 is considerably more shallow than the slopes for the other trials. This is certainly due to a practice effect on the second trials. This slope change between first and second trials is not apparent for Group 4 Trial 2 who did an SIC task, not shown graphically, first and then the TIC No Bell task. They

G1T1 No Bell TIC Task of 1st Trial
G1T2 Bell TIC Task on 2nd Trial
G2T1 Bell TIC Task on 1st Trial
G2T2 No Bell TIC Task on 2nd Trial
G3T1 No Bell TIC Task on 1st Trial
G4T2 No Bell TIC Task on 2nd Trial



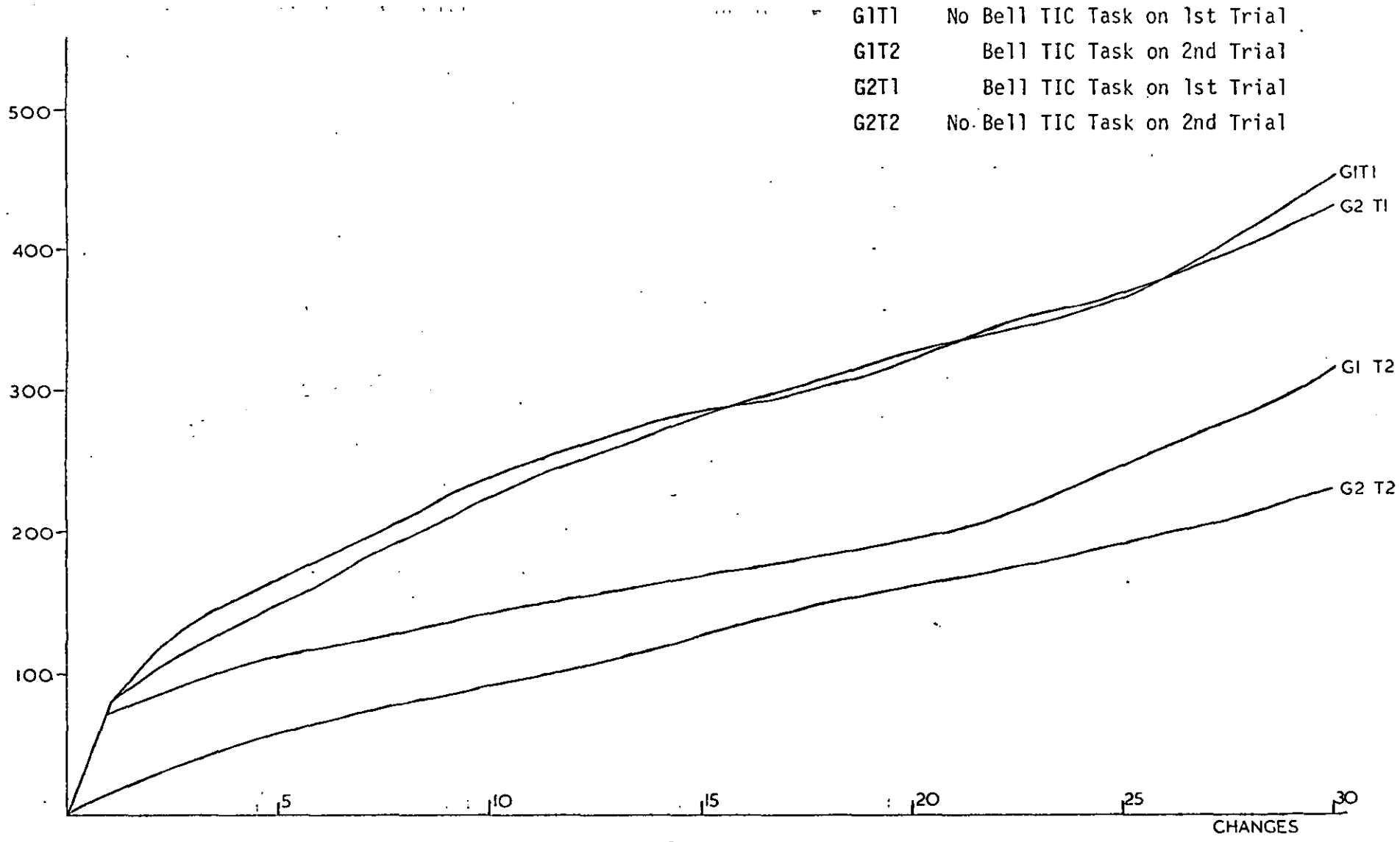
MEAN ABSOLUTE CUSUM OF ERRORS

Fig 39

show very little difference over the groups who did the No Bell task first indicating no positive transfer of training between the SIC and TIC task.

These errors have been plotted against time and do not include the error scores of the SIC groups. This is because the subjects doing the SIC task had a different objective from the subjects in the TIC task. Whereas the subjects in the TIC task were asked to keep the error score at minimum all the time, the subjects doing the SIC tasks were only told to "get each stage as good as possible before changing". They were not worried about large overloads or underloads occurring between changes. However both groups did have one objective in common; they were both required to have the best possible optimisation just before a change occurred. In order to fully compare the 2 tasks the Mean Absolute Cusums were replotted against changes, as opposed to time. The graphs are shown in Fig 40 for Experiment 1 and Fig 41 for Experiment 2.

Considering Fig 40 first there is virtually no difference in these plots compared with the plots against time of Fig 39 indicating that it is reasonable to consider the errors when plotted against changes. In the graphs for Experiment 2 in Fig 41, the plot for Group 3 Trial 1 is the same as it was in Fig 39, as is the plot for Group 4 Trial 2; this again supports the validity of plotting against changes as opposed to time. The 2 plots for the SIC tasks Group 4 Trial 1 and Group 3 Trial 2 show a dramatic difference in shape compared with the TIC tasks. The slope is shallower during the initial stages and slopes at the end for the SIC tasks compared with the TIC tasks. This is certainly due to the subject taking considerably more care during the initial stages of the SIC task, this takes very much more time than the even paced schedule



MEAN ABSOLUTE CUSUM OF ERRORS

Fig 40

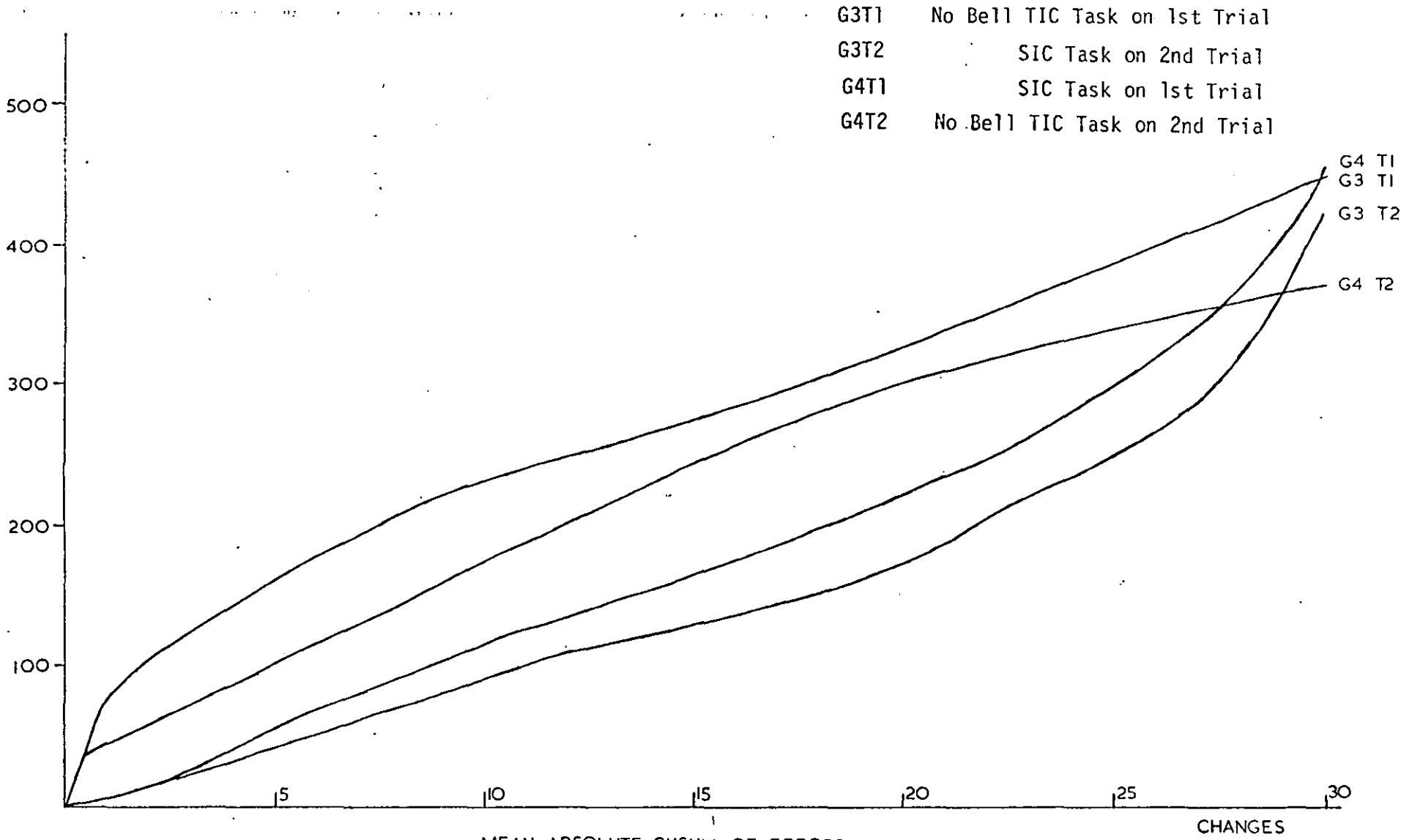

G3T1 No Bell TIC Task on 1st Trial
G3T2 SIC Task on 2nd Trial
G4T1 SIC Task on 1st Trial
G4T2 No Bell TIC Task on 2nd Trial
500
400
300
200
100
G4 TI
G3 TI
G3 T2
G4 T2
5
10
15
20
25
30
CHANGES
MEAN ABSOLUTE CUSUM OF ERRORS

Fig 41

of one minute would allow. Some subjects waited 6 minutes before making the first change. This meant that towards the end of the trial they were pushed for time and had to rush the last optimisations giving much greater errors towards the end of the trial.

The graphs in Figs 40 and 41 are in fact the mean of the group of subjects. The actual individual scores are shown in Annex D Figs 1-8. Annex E shows the individual plots by groups of the relative cumulative sums for the 2 experiments. These graphs are much more difficult to interpret than the absolute cusum graphs; however it can be seen that the subjects erred on the side of underloading rather than on the side of overloading. This is due to the fact that the instructions stated that errors of overloading were considered to be twice as bad as errors of underloading. It does not appear that subjects were taking the instructions as literally as they might have done. The errors of underloading tolerated are considerably more than twice the value of the errors of overloading that were tolerated.

3.5 Command usage

As part of the command sequence analysis the actual usage of individual commands utilised in terms of frequency of occurrence. For the command ON. there were significantly more for Group 3 Trial 1 and Group 4 Trial 2; however I believe this to be due to exceptionally high scores attributable to 2 separate subjects rather than an experimental effect. For the command OFF the same effect on Group 3 Trial 1 and Group 4 Trial 2 occurs due to the high individual usage of 2 subjects. There is no significant difference $p = .47$ between the ON and OFF commands across groups; this is to be expected. For the command SUPPLY there was significantly less usage $p < 0.01$, of SUPPLY in the No Bell conditions in both experiments compared with the Bell and SIC tasks.

For the command STATUS there is an exceptionally low number of STATUS commands for the SIC condition as opposed to the other conditions. This is primarily due to the use of the all OFF strategy within the SIC tasks, see Section 3.2.5, the subjects not then needing status information. For the command DEMAND no across group pattern emerges but it is interesting to note that one subject used the DEMAND command 49 times whereas the mean for the subjects was 20. For the command SCORE it is not possible to consider the SIC task as there were exactly 30 changes for each subject. For the TIC task there were significantly more demands SCORE in the No Bell condition than in the Bell condition ($p<0.01$); this is due to the need to monitor using the SCORE command in the No Bell task. For the command ADD there was an exceptionally high number of commands ADD issued in Group 4 Trial 1, the SIC task on the first trial, in fact more commands ADD than all other conditions added together. The use of the command ADD when the SIC task was done second, Group 3 Trial 2, was not significantly different from all other conditions. This discrepancy between the 2 SIC conditions is almost certainly due to a transfer of training effect where it was not felt necessary to use the ADD command to assist with the arithmetic after having done the TIC task, but without having done the TIC task the SIC subjects thought it useful to use. This result was not due to one subject but was spread evenly over all 4 subjects. The command usage shows considerable subject individual difference and also, for certain commands, differences between groups and trials.

3.6 Interrupts

As described earlier the subjects had the ability to interrupt the computer's output whenever they chose. The commands that it was possible to interrupt were the listing commands; SUPPLY, STATUS, DEMAND, SCORE

and ADD. Of these only 2 were ever interrupted, these were STATUS and DEMAND, which are the 2 commands producing the longest lists. STATUS was interrupted only very occasionally and no conclusions can be drawn from such little data. The command DEMAND was however interrupted fairly frequently. Two experimental groups used more than 80% of all interrupts; Group 4 Trial 1, the SIC task first, used 50% of all the interrupts to the command DEMAND and Group 2 Trial 1 the TIC with Bell task first,used another 30% of the total interrupts. These results were not due to individual scores but were evenly distributed acorss all subjects.

I believe these results can be explained in 2 parts; firstly the large usage of interrupts only occurs on the first trial, so I presume the subjects to be using this interrupt facility to keep up with the tasks; and after they had settled down and found the 'pace' of the experiments they did not need to use interrupts so often. Secondly, these large number of interrupts was only used on the tasks where the subject knew a change had occurred and suggests that subjects were stopping the output when they saw the change that they knew had occurred. There is obviously a need to allow the user to interrupt a computer's output, especially when he has seen what he wants or when he has entered an erroneous command.

3.7 Questionnaire

Subjects were given 2 questionnaires by interview. The first before the experiment was designed to determine their experience with computers; they were asked how many times they had used a teletype or interacted with a computer, what languages they knew, and their general experience of computers. They were ranked by the experimenter in his opinion as to

their experience. These rankings were compared with their error performance on each trial; no significant correlation was detected with Spearmans rank correlation coefficient on either of the trials; ($r = .3$ for first trial and $r = .5$ for second trial).

After the experiement the subjects were asked questions regarding the experiment.

They generally had no difficulty understanding the functions of the command or their application and had no difficulty with the command system. When asked whether the changes in the TIC task were regular or irregular, most subjects (94%) said they were irregular. The subjects were asked what they thought the average change interval was and stated a range 3-5 minutes between changes, the actual being exactly 1 minute all the time. An open ended question was asked regarding the strategies employed in order to gain further evidence to support possible solution algorithms. Subjects were also asked which of the 2 tasks they found easier. For Experiment 1, 3 subjects said they found the No Bell task easier and 5 said they found the Bell task easier. For Experiment 2, 4 subjects preferred the SIC task and 4 the TIC task. There is no apparent interaction between performance and Task Order.

3.8 Aspiration Analysis

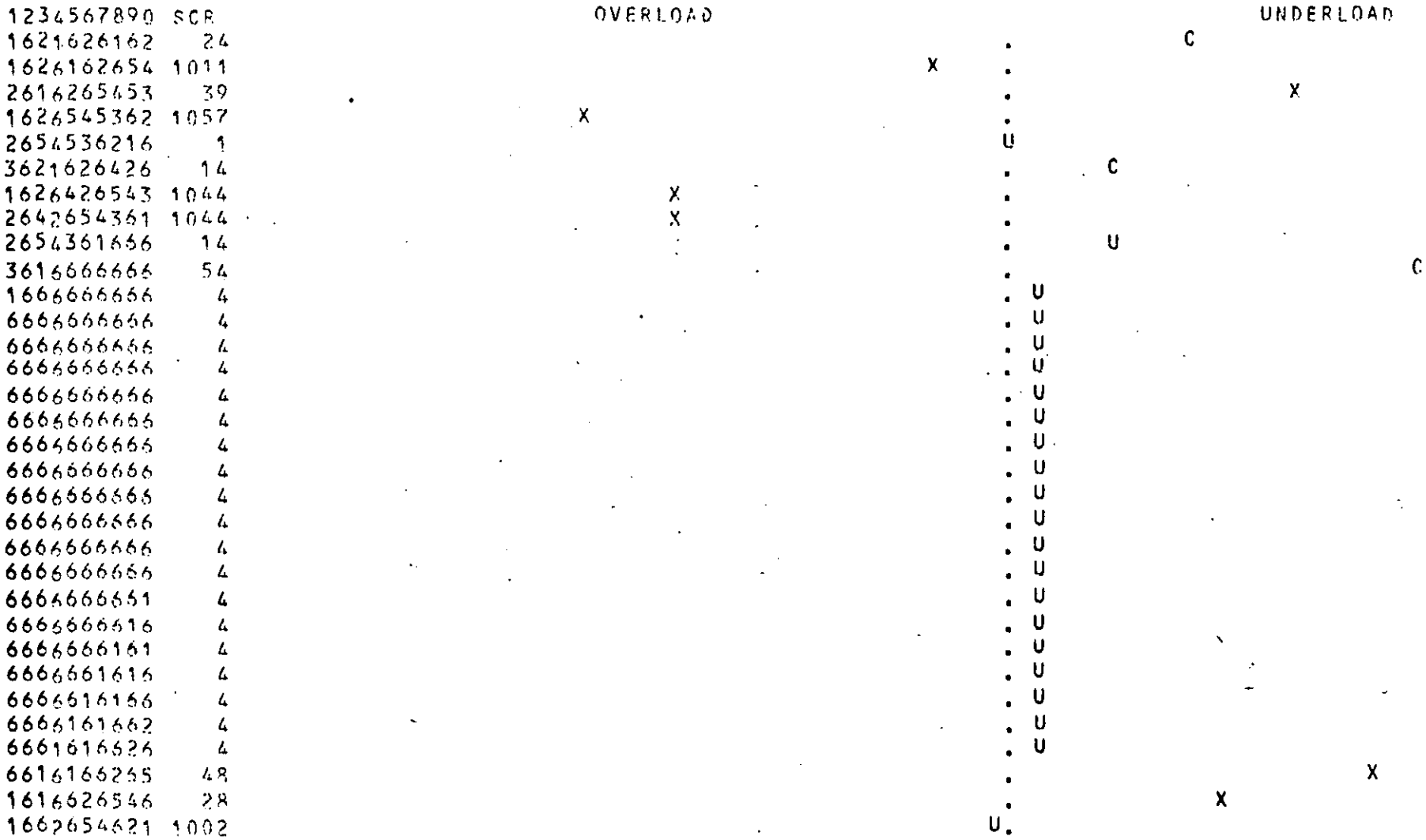
Taking the statements made during the command sequence analysis a step further. It would appear that the sequence of commands issued and their relation to the error score at that time should determine which tactics the subjects adopted for the next period. It is suggested that if the subjects had a tolerable error score they would not change anything. If the error score was large, then they would go through a process of detection and correction of errors. If however the error score was enormous they would take some emergency corrective action.

Figs 42 and 43 show a computer analysis for 2 subjects on individual trials. On the far left is the sequence of commands issued. The 6 in column 2 is the SCORE command to which the subjects received the value of error shown in the column after the command sequence. For this score value the command before the score command in column 2 and the 8 subsequent commands make up the command sequence. The commands are plotted down the page for each SCORE command issued. To the right is a graph of the analogue value of the score command. The centre line is zero error, to the left is overloaded to the right is underloaded. The scores greater than 1000 indicate an error of overloading of value equal to score minus 1000.

An algorithm was developed to plot a C in the correct analogue position for the score when the commands issued after the SCORE command were either STATUS, SUPPLY or DEMAND followed by ON or OFF, indicating some data collection and a calculated change. The computer plots a 'U' for unchanged when the command following a SCORE before the next SCORE did not include an ON or OFF command. It plotted an X for emergency change when the commands immediately following the SCORE were ON or OFF.

This algorithm was developed over a number of iterations and is at this time the most likely simple representation of the subjects decision making. It was hoped that it would be possible to draw on the graph the line determining the thresholds of toleration of error or aspiration of the subjects, with all the 'U's near the centre, the 'C's towards the outside, and the 'X's at the extreme. However it can be seen that this did not entirely work. The main problems in this type of analysis are that subjects make errors, that not all information is

ASPIRATION ANALYSIS FOR SUBJECT 1 OF GROUP 1 ON TRIAL 1



Fig·42

ASPIRATION ANALYSIS FOR SUBJECT 15 OF GROUP 3 ON TRIAL 2

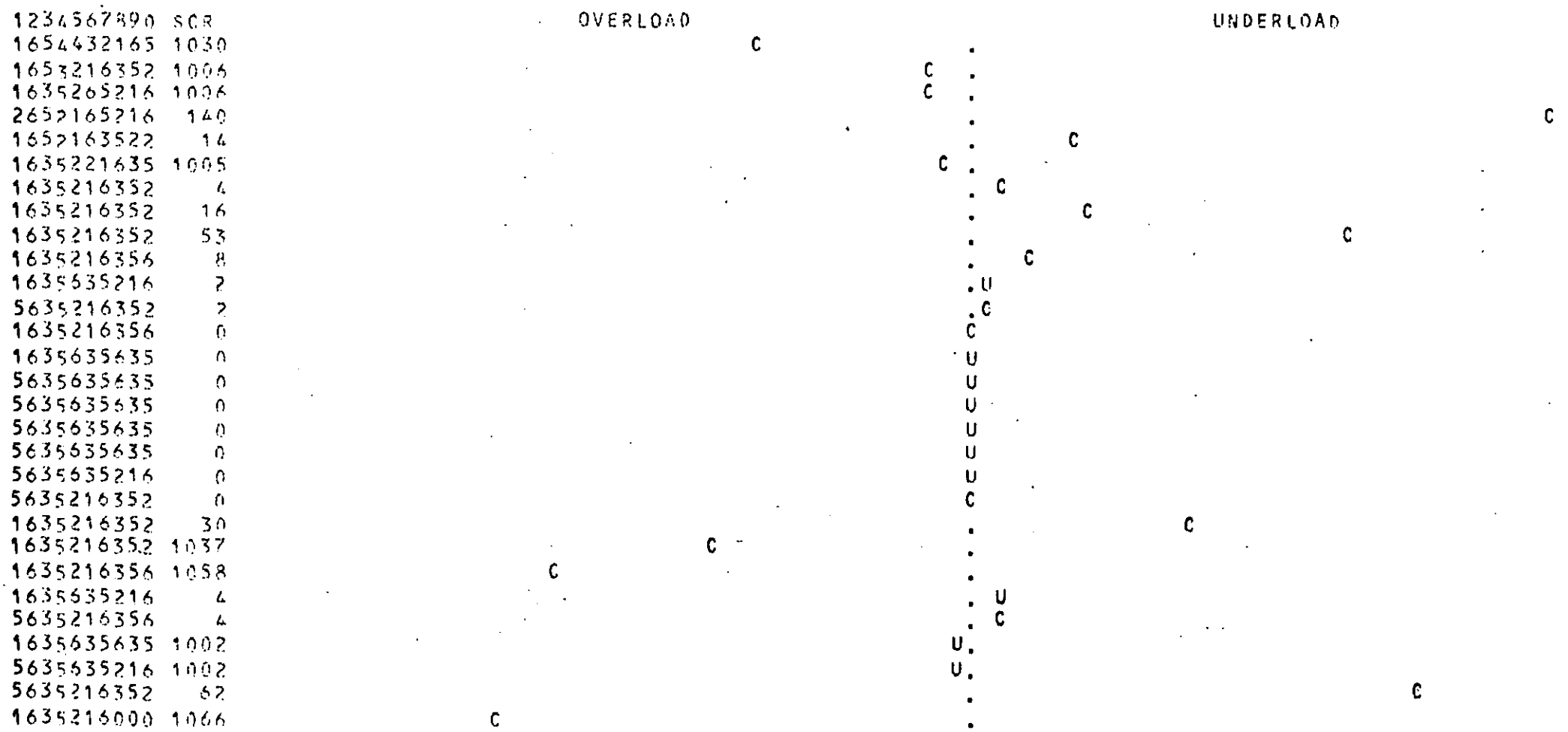


Fig 43

recorded, and that the command sequence is time related but this is not reflected in this analysis. This sort of analysis could yield valuable insight into the decision making processes if these problems can be resolved.

3.9 Summary of Results

This summary identifies from the preceding sections those results having most significance. It makes no attempt at explaining or relating the different measures, this is done during the discussions in Part 4.

3.9.1 Experiment 1

1. In the Analysis of Variance for all task conditions there was a significant difference in task Wait and Input times due to subject differences.
2. Fig 6 shows a significant interaction between Task and Task Order due to a much shorter Wait time for the No Bell task when it was done second compared with when it was done first, and no similar difference in the Bell Wait times.
3. Fig 7 shows the command OFF having a much longer Wait time than other commands.
4. Fig 8 shows the effect noted in 3 above to be task dependent and that No Bell task OFF Wait time was longer than the Bell task OFF Wait time.
5. Fig 11 shows the commands OFF, ON and ADD have a significantly longer Input time than other commands.
6. Fig 16 shows a decrease in Wait and Input time over the experiment.
7. In the command sequence analysis the No Bell tasks had many occasions during which the SCORE command was repeated,

8. In the cumulative error analysis the errors on the first task were nearly equal, regardless of which task was done first, Fig 40.

9. Fig 40 also shows the second task error scores to be better than the first task.

10. The best error scores were achieved by the group who did the No Bell task second.

3.9.2 Experiment 2

1. In the Analysis of Variance for all task conditions there was a significant difference in task Wait and Input times due to subject differences.

2. The mean Wait time for the SIC task was almost twice the mean Wait time for the TIC task, Fig 18.

3. The commands OFF and ADD had significantly longer Wait times than other commands, Fig 19.

4. The ADD command on the SIC task Wait time was significantly longer than the other commands, when it was done first, Figs 20 and 21.

5. Fig 23 shows the commands OFF, ON and ADD to have had significantly longer Input times.

6. Fig 24 shows a reduction in Input time as the experiment progressed.

7. There was a difference in the ON and OFF Input times for the SIC task that was not apparent for the TIC task, Fig 25.

8. The ADD command had a much longer Input time in the SIC task than the TIC task, Fig 25.

9. The command sequence analysis shows much more cyclic command sequences in the SIC task than the TIC task, Figs 33 and 34.

10. The command STATUS was not used nearly as much in the SIC task as in the TIC task.

11. The sequence SCORE, ON or OFF, SCORE never occurred in the SIC task.

12. The cumulative error curves have a different shape for the SIC task than the TIC tasks, Fig 41.

3.9.3 Across Experiments

1. The command DEMAND was interrupted often mainly by Groups 4 Trial 1 and Groups 2 Trial 1.

2. Subjects were divided evenly in their preferences for task conditions.

PART 4

DISCUSSIONS AND CONCLUSIONS

PART 4 CONTENTS

4.1 Discussions

4.1.1 Experiment 1

The object of Experiment 1 was to determine the effect of information concerning the occurrence of a change in the state of the system on the subjects performance at an optimisation task. It is clear from the data that there is an effect specifically due to the presence or absence of this feedback information. When the subjects were not given feedback information they had no means of knowing when, or if, a change had occurred, without examining the system. Consequently they adopted a monitoring strategy designed to detect a change when it occurred. Their strategy was in fact slightly more sophisticated and was designed to show not only when any change occurred, but when a change occurred that affected their optimisation. Their strategy was to simply type SCORE, hence demanding feedback, at regular intervals evidence that this strategy occurred is given in Figs 31-34 where long periods can be seen when only the command SCORE was used, these periods coinciding with periods of negligible error, as shown in Figs 35-38. Substantiating evidence is given by the command usage analysis showing many more commands SCORE in the No Bell task than in the Bell task.

I believe there is evidence to show that the adoption of this monitoring strategy caused a decrement in overall task performance; this can be shown with reference to a tactic used by the subjects to assist them doing the task. It is evident from the data in Fig 7 that the command OFF had a much longer Wait time than any other commands. The post-test questionnaire indicated that this was due to the subjects, when they needed to re-allocate areas, working out the complete optimisation then selecting those areas they needed OFF first, before selecting ON those areas needed to complete the re-allocation. This tactic was also

necessary in order to comply with the instructions, that errors of overloading were worse than errors of underloading. Fig 8 shows the effect of Fig 7 to be task dependent. It shows that the larger portion of the Wait time for the OFF command was used in the No Bell condition. I suggest this shows a penalty due to the adoption of the monitoring tactic. When the subjects could wait for the Bell to ring they used this time to consolidate their position and hence needed to wait a shorter time before commencing a re-allocation than those subjects having No Bell who could not consolidate their position to the same extent.

The evidence from the error scores does not support this hypothesis, in fact it may be seen as contradicting it. Fig 40 shows that on the first trial the error scores are nearly identical regardless of task, and that both the second trials are better than the first trials. It also shows that the error scores for the No Bell task when done second are LESS than the error scores for the Bell task when it is done second, contrary perhaps to expectation. However I suggest that this is evidence of an 'expected difficulty' phenomenon. The subjects who did the No Bell task first expected their second trial, with the Bell, to be easier; and the subjects having done the Bell task first concentrated harder when expecting no assistance from the Bell. This is supported by the observation that the majority of the difference is in the very early stages before they had time to settle down to the tasks. I suggest that the error score evidence is inconclusive when considering the monitoring hypothesis.

Within Experiment 1 there is evidence not contributing directly to the feedback problem that is nevertheless important. The observed Task-Task Order interaction shown in Fig 6 is very difficult to explain.

The effect stated again is this:

1. The mean Wait time for 4 subjects on the Bell task, across Commands and Periods, is unaffected by Task Order.
2. The mean Wait time for 4 subjects when the No Bell task is done first, across Commands and Periods, is significantly longer than the Bell task.
3. The mean Wait time for 4 subjects when the No Bell task is done second, across Commands and Periods, is significantly shorter than the Bell task.

The most reasonable explanation I can offer is as follows. In both Task Orders there was a reduction in the mean Wait time on the second trial from the first trial of approximately 1 second. Combined with this the subjects doing the Bell task on the first trial started off with a lower mean Wait time than the subjects doing the No Bell task first. This would indicate the presence of Bell feedback having a marked effect on the first trials.

From these results I would conclude that there is evidence to show both tactical differences in the approach subjects had to the tasks, due to feedback concerning the occurrence of a change of state in the system but that the evidence is confounded to a certain extent by the effects of the task itself and the problem solving strategy adopted by the subjects.

4.1.2 Experiment 2

The object of Experiment 2 was to study the effects of different time structures on identical tasks. It is clear from the data that the different time structures elicit different tactics and strategy from the subjects and produce real performanc differences. It is clear from the data that the group doing the TIC task first are in no

respect significantly different from the group doing the No Bell TIC task first in Experiment 1, providing a useful validation of some of the Experiment 1 effects.

The most dramatic difference between the tasks is shown in the Task effect of Fig 18. This shows that the mean Wait time, across Subjects, Commands and Periods, was twice as long for the SIC task as for the TIC task. This is certainly due to the subjects in the SIC task deliberating much longer than the TIC subjects before issuing a command and partly due to the effect of the monitoring strategy used in the TIC task, the subjects using many SCORE commands with short Wait times. This hypothesis is substantiated by the evidence from the command sequence analysis showing the sequence SCORE, ON or OFF, SCORE, indicating action taken in a hurry without data, which never occurred in the SIC task.

The subjects used different tactics in their approach to the 2 tasks; in both tasks they selected areas OFF before re-selecting areas ON. In the SIC task they turned ALL the areas OFF by using the default option to the command, no areas in the parameter list. This was not a permissible tactic to use in the TIC, as it caused massive errors of underloading during the period before areas were re-selected ON. Evidence to support these observations can be found in the command usage analysis which found a significantly lower number of occasions on which the command STATUS was used. If they turned ALL areas OFF they knew exactly what their status was. Evidence is also available in Fig 25 which shows a significant reduction in the Input time for the OFF command compared with the ON command, showing that fewer parameters were used, usually none. The same graph for the TIC task

shows no difference in the Input time for the ON and OFF command, indicating that both commands were paramterised to the same extent.

The cumulative error analysis shows considerable effects due to the differences in tasks. The most obvious feature of Fig 41 is the totally different shape of the TIC curves from the SIC curves. This shows the TIC curves, Group 4 Trial 2 and Group 3 Trial 1, have a shape similar to the cumulative error scores in Experiment 1, with a high initial slope and a gradual reduction in slop as the experiment progresses. The SIC curves, Group 4 Trial 1 and Group 3 Trial 2, however have a shallow initial slope which gets increasingly sharper towards the end of the experiment. I suggest this is because the subjects took much more care in the early stages of the SIC task than the TIC task subjects were able to, but that this extra care meant they had to take much longer over each optimisation than an even paced schedule of optimisations would allow. This made them short of time at the end and they had to rush to get the experiment finished, producing more errors. Fig 41 also shows evidence in differential transfer of training between tasks. The subjects who did the SIC task second had a better error performance than those who did the SIC task first showing some positive transfer of training between TIC and SIC tasks. On the other hand subjects who did the No Bell task second, having done the SIC task first did not do as well on the second trial as those subjects in Experiment 1 on their second trial. This shows less positive transfer of training between SIC and TIC than there was between TIC No Bell and TIC Bell in Experiment 1.

The indications in the previous paragraph that the subjects in the SIC task were trying too hard is supported by the data shown in Figs 20 and

21. These figures show an extremely complex interaction centred around the ADD command. Fig 20 shows an extremely long Wait time for the ADD command in the SIC task compared with the TIC task. Fig 21 shows that this Wait time was longest when the SIC task was done first, this corresponds to Group 4 Trial 1.

If the ADD command, a command designed to make the tasks easier, should require such a long Wait time this must be interpreted as the subjects trying too hard to get a perfect optimisation.

4.1.3 Both Experiments

There were 3 other findings regarding task performance that were common to both experiments and reflect the nature of the task and its implementation.

The subjects had no training before these experiments and they would be expected to show a practice effect. This did occur and the evidence can be seen in Figs 12, 16 and 24, all showing a reduction in the time before a command was issued as the tasks progressed.

Another significant finding of this thesis contradicts the findings of WEINBERG (1972). In studying the effects of parameters added to command words he found that adding parameters to commands increased the speed of entry by 30% over single commands but produced 3 times the number of errors. The evidence of Figs 11 and 23 shows an increase in entry time of approximately 3 times when a command had to be parameterised. This does not mean WEINBERG was wrong but more likely that parameterisation is command and task specific in its effects.

An important finding of this thesis, not related at all to the tasks under discussion, concerned the use of the interrupt facility. The facility to interrupt the computers output at any time by pushing any key was included in the task implementation simply because the author, in his role as an experienced computer user, was frustrated when, having issued an erroneous command, he could not stop the computer from completing the command. The command was used extensively by the naive subjects who did the task but only on one command, DEMAND; and mainly by 2 groups, the group doing the SIC task first and the group doing the Bell task first. These findings must be confirmed and explained by further research, but the message to all computer designers is clear. If the use of an interrupt would not produce an irrecoverable error, the user should have the facility to over-ride the output and shut the computer up.

4.1.4 Measures

The measure of Wait and Input response times analysed with Analysis of Variance techniques proved to be a very powerful method of establishing differences in task performances. There is apparently no need to combine the Input and Wait times together, no useful additional information being obtained. The Analysis of Variance gave a good insight to the complex interactions that are inevitable in such tasks as these.

The command sequence analysis as described was disappointing. The promise showed by the graphs to distinguish patterns of activity was not carried on to provide an absolutely defined measure. Considerably more work needs to be done on such sequence analysis before this type of analysis can yield the information that is so obviously there.

The error scores, particularly the way the error was calculated, proved to be an attractive measure. However it will not always be possible in future experiments to judge the number of better ways a subject could have performed. This may make the error scores more dependent on the tasks, and numbers involved in the task, than those reported here. The principle of cumulative sum error scores seems to be by far the best way of presenting and interpreting this sort of error data.

The command usage analysis was not expected to prove particularly dramatic and indeed these expectations were fulfilled; nevertheless it gives a useful indication of the tactics adopted by the subjects. In fact the first indication of the all OFF tactic was from the command usage analysis, by showing that some subjects never used the STATUS command.

The aspiration analysis shows great promise but at this stage it is impossible to tell whether the promise will come to anything.

4.1.5 Tasks

The main problem facing the author and future experimenters in this field is the problem of the degree of dependence the answers have on the specific task studied. If the object of the experiments is to generalise investigatory research then the task used must be kept as simple and as straightforward as possible. I believe the goal of a universal standardised task for studying man-computer systems in all specific situations is unobtainable. This means that the 3 factors of controllability, measurability and realism have different weights depending on the aim of the experiment.

If the aim of the experiment is, as in this thesis, general investigatory research then controllability is of prime importance; however if the aim of the experiments is to solve task specific problems, then the prime requirement must be for realism. These 2 conflicting requirements cannot be met by one generalised task.

What can be said however is that the resource allocation task embodies many of the features present in a large group of man-machine system problems and as such is a useful task environment for future study.

4.2 Conclusions

There were 2 areas of this thesis that can, in retrospect, be seen to be mistakes. The first was the inadequate control of the tasks complexity and the second was the optimistic proposition that problem solving strategies could be predicted by simple models.

Task complexity was inadequately conceived and implemented; there is obviously a need to control and measure the difficulty of the task but the scheme used was not satisfactory. It can be seen from the error score curves that the errors are more or less constant even though the task was allegedly getting more difficult. In future experiments of this type it must be possible to extract the variations due to task difficulty from the performance data to prevent confounding, as occurred in these experiments.

The proposed models of possible solution algorithms, whilst convenient in theory, were just too simple to predict the complex behaviour of the subjects. They must be greatly refined before they are any use to man-computer interaction research.

At a detailed level there were 2 significant findings from the thesis concerning parameterisations and interruptions. The apparent contradiction between the results of WEINBERG (1972) and those presented in this thesis give rise to thought and must be resolved by further experimentation, but it is possible that the contradiction occurs either because the results are task specific or because our frames of reference arenot coincident. The observations and results concerning the interruptions of the computers output have significant implications for the systems designers but more research, specifically directed in this area, is advisable before definite recommendations can be given.

There are 2 main findings at a cognitive level, the first concerning feedback and the second concerning time. The provision of feedback to the subjects when the system changes its state could be regarded as augmented knowledge of results;if this is the case these results are in agreement with those reported by ANNETT (1969). The addition of feedback does 'improve' performance but the removal of feedback after training did not adversely affect performance.

Perhaps the most important finding of these experiments is in the relationship between tasks and time. Whilst it was expected that different time structures would produce different tactics the differences were not expected to be so marked. This raises serious questions as to the ability of man to alter his strategies to cope with different time emphases, and to the desirability or otherwise of changing task time relationships. It would appear that some subjects liked to be paced by the machine whilst other preferred to go at their own pace. This has implications for selection and training. Do people have different capabilities under different time

bases? If this experiment identifies 2 types of people how many more are there? Is time the most fundamental variable in man machine systems?

I believe that this research has shown the approach and methodology described in this thesis to be viable in determining some of the problems of man-computer interaction. It is possible to pursue investigatory research with such a task, and control it to a sufficient degree, so that useful answers are obtained. I suggest this has not been achieved previously.

Undoubtedly the success or failure of such research depends on a successful balance between realism, controllability and measurability. To do pure investigatory research I believe it would be unwise to attempt to make the tasks any more realistic than the simple one used here as it may become uncontrollable. However in a task specific problem situation a large amount of controllability could be sacrificed in the pursuit of realism without commensurate loss of validity, but the results would then also be task specific and would not generalise to other situations.

Before it is possible to measure the efficiency of man-computer interaction it is necessary to do further research in order to identify and control all relevant variables, to understand when a simulation is a realistic representation of the real world, and to quantify and aggregate system performance measures into one single benefit term. I believe this thesis is one small step towards the achievement of these objectives.

PART 5

REFERENCES

5.1 References

ANNETT J 1969 Feedback and human behaviour. Penguin London.

BENNETT R K 1969 The design of computer languages and software systems; A basic approach. Computers and Automation. V18(2) pp 28-33.

BERKELEY E C 1969 The personality of the interactive programmed computer. Computers and Automation. V18(13) pp 42-46.

BERNSTEIN W A and OWENS J I 1968 Debugging in a time-sharing environment. American Federation of Information Processing Societies. Fall Joint Computer Conference. V38(1) pp 7-14.

BOBROW D G 1967 Problems in natural language communications with computers. IEEE Transactions on Human Factors in Electronics. HFE-8(1) pp 52-55.

BOEHM B W, SEVEN M J and WATSON R A 1971 Interactive problem solving - an experimental study of lockout effects AFIPS V38 pp 205-210.

CARBONELL J R, ELKIND J I and NICKERSON R S 1968 On the psychological importance of time in a time sharing system. Human Factors. V10(2) pp 135-142.

CHAPIN N 1965 What choice of programming languages? Computers and Automation V14(2) pp 12-14.

CHASEN S M and SEITZ R N 1967 On line system and man-computer graphics. Computers and Automation V16 November pp 22-28.

COONS S A 1966 Computer graphics and innovative engineering design. Datamation May pp 32-34.

CORNEY G M and TAYLOR I M 1970 Experiments in input formatting. EMI Electronics Ltd Ergonomics Memo 268.

DAVIS R M 1966 Man-Machine communication. In annual review of Information Science and Technology Vol 1 Wiley New York pp 221-254.

FRASER J B 1967 On the role of natural language in man-machine communication. In Information System Science and Technology Thompson Washington.

GOLD M 1967 Methodology for evaluating time shared computer usage. Doctoral Dissertation. MIT Alfred P Sloan School of management. Mass.

GRISHMAN R 1970 The debugging system AIDS. AFIPS. FJCC V36 pp 59-64.

HALPERN M 1965 Computer programming: The debugging epoch opens. Computers and Automation. V16 November pp 28-31.

HAYS W L 1969 Statistics. Holt, Rinehart and Winston, London.

JONES G M, HUGHES J L and ENGVOLD K H 1970 A Comparative study of management decision making from computer terminals. AFIPS SJCC V36 pp 599-605.

LICKLIDER J C R 1965a Man-Computer partnership. International Journal of science and technology May pp 18-26.

LICKLIDER J C R 1965b Languages for specialisation and application of prepared procedures. Information systems sicences. Proc of Ind Congress. Spartan Books pp 177-187.

MILLER R B 1968 Response time in man-computer conversational transactions. AFIPS FJCC V33(1) pp 267-277.

MILLS R G 1967 Man-machine communication and problem solving. Annual review of information science and technology V2 pp 223-253.

MORRIS C 1938 Foundation of the theory of signs. An international encyclopedia of unified science. V1(2) University of Chicago.

NEISSER U 1965 MAC and its users. Massachusetts Institute of Technology, Cambridge, Mass. Project MAC Memo 185.

NICKERSON R S 1969 Man-computer interaction: a challenge for human factors research. Ergonomics 12(4) pp 501-517.

NICKERSON R S, ELKIND J I and CARBONELL J R 1968 Human factors and the design of time sharing computer systems. Human Factors 10(2) pp 127-134.

NICKERSON R S and PEW R W 1971 Oblique steps toward human factors engineering of interactive computer systems ARPA Contract No F44620-71-C-0065

PARSONS H M 1970 The scope of human factors in computer based data processing systems. Human factors V12(2) pp 165-175.

RAPHAEL B 1966 The structure of programming languages. Communication of the ACM V9(2) pp 67-71.

ROOT R T and SADACCA R 1967 Man computer communication techniques; two experiments. Human factors V9 pp 521-528.

SACKMAN H 1967 Computers, System Science and Evolving Society Wiley New York

SAMMET J E 1966 The use of English as a programming language. Communications of the ACM V9(3) March pp 228-230.

SAMMET J E 1969 Programming languages; History and fundamentals Prentice Hall New Jersey.

SCHEFFE H 1959 The analysis of variance Wiley New York.

SCHERR A L 1966 Time sharing measurement. Datamation V12(4) April pp 22-26.

SCHRENK L P 1969 Aiding the decision maker - A decision process model Ergonomics V12 pp 543-558.

SCHWARTZ J I 1965 Comparing programming languages. Computers and Automation V14(2) pp 15-16, 26.

SCHWARTZ J I 1968 Interactive systems - promises, present and future AFIPS FJCC V33(1) pp 89-98.

SIEGEL S 1959 Non-parametric statistics for the behavioural sciences. McGrawhill

SIMON H A 1966 Reflections on time sharing from a users point of view. Computer Science Research Review. Carnegie Institute of technology.

SHACKEL B 1969 Man-computer interaction: The contribution of the human sciences. Ergonomics V12(4) pp 485-499.

TEITELMAN W 1969 Toward a programming laboratory. Bolt, Beranek & Newman, Inc Cambridge, Mass.

WEINBERG G M 1972 The psychology of computer programming. Van Nostrand Runhold New York

WINER B J 1962 Statistical principles in experimental design McGrawhill New York

YNGVE V H and SAMMET J E 1963 Toward better documentation of programming languages. Communications of the ACM V6(3) March pp 76-92.

ZEMANEK H 1966 Semiotics and programming languages. Communications of the ACM V9(3) March pp 139-143.

ANNEX CONTENTS

ANNEX A

TASK SOFTWARE

```
C
C       MAINRB
C
C       R.G.ASHTON      HUSAT   LOUGHBOROUGH UNIVERSITY         1972
C
C       THIS IS THE MAIN PROGRAM FOR THE REAL TIME WITH BELL "IRATE" TAB
C       IT REQUIRES THE PROGRAM 'START' TO BE LOADED WITH IT.
C
        COMMON I10,ISUP,IP,IS,IL,MESS,ITIME,IRAND,ISUB,IGRP,IRUN,X1
        DIMENSION IP(7),IS(7),IL(7),MESS(10,10),ICOM(250),IRAND(2,29),
       1 NTIME(750),ITOT(250)
C       ZERO ARRAYS THAT HOLD DATA
        DO 1000 I=1,250
        ICOM(I)=0
1000    CONTINUE
        DO 1001 I=1,750
        NTIME(I)=0
1001    CONTINUE
        DO 1002 I=1,250
1002    ITOT(I)=0
C       ZERO INDICES
        INA=0
        INB=0
        INC=0
        IND=0
        I10=0
C       CALL SET UP ROUTINE
        CALL START
C       THIS PART OF THE PROGRAM CONTROLS THE ROUTING OF THE PROGRAM
C       BEFORE AND AFTER A COMMAND IS COMPLETED. TEST FOR MORE THAN
C       250 COMMANDS, IF SO FINISH.
90      IC=0
        IF(INA-750) 9,9,8
8       CONTINUE
S               JMS FINISH
9       CONTINUE
C       GET COMMAND
S               JMS COMAND
S               DCA \N
        IC=0
        IE=0
C       THE ROUTINGS FOR COMMANDS ARE GIVEN BELOW
C            ON OF SU ST DE SC AD
        GOTO(10,11,30,40,50,60,70) N
C       TO 10 FOR COMMAND ON, TO 11 FOR COMMAND OFF
10      L=1
        GOTO 20
11      L=0
20      I=0
21      I=I+1
        IF(I-8) 22,90,90
22      K=IP(I)
        IF(K) 90,90,23
23      IS(K)=L
        GOTO 21
C
C
C
```

```
C       TO 30 FOR COMMAND SUPPLY
30      CONTINUE
S               CLA;TAD (5      /MESSAGE 5
S               JMS CWRITE
S               TAD \ISUP       /SUPPLY
S               JMS IWRITE
S               TAD (6          /MESSAGE 6
S               JMS CWRITE
S               JMP \90
C       TO 40 FOR COMMAND STATUS
40      I=0
41      I=I+1
        IF(I-8) 42,90,90
42      K=IP(I)
        IF(K) 90,90,43
43      CONTINUE
S               CLA;TAD (4      /MESSAGE 4
S               JMS CWRITE
S               TAD \K          /AREA
S               JMS IWRITE
        L=IS(K)
S               CLA;TAD \L      /STATUS OF SWITCH IS(K)
S               SZA;JMP L44
S               CLA;TAD (2      /MESSAGE 2
S               JMS CWRITE
S               JMP \41
S       L44,    CLA;TAD (1      /MESSAGE 1
S               JMS CWRITE
S               JMP \41
C       TO 50 FOR COMMAND DEMAND
50      I=0
51      I=I+1
        IF(I-8) 52,90,90
52      K=IP(I)
        IF(K) 90,90,53
53      CONTINUE
S               CLA;TAD (3      /MESSAGE 3
S               JMS CWRITE
S               TAD \K          /AREA
S               JMS IWRITE
        L=IL(K)
S               CLA;TAD \L      /DEMAND VALUE
S               JMS IWRITE
S               TAD (6          /MESSAGE 6
S               JMS CWRITE
S               JMP \51
```

```
C       TO 60 FOR COMMAND SCORE
60      IADD=0
        DO 62 I=1,7
        IF(IS(I)) 62,62,61
61      IADD=IADD+IL(I)
62      CONTINUE
S               CLA;TAD (D8     /MESSAGE 8
S               JMS CWRITE
        J=ISUP-IADD
        I=IABS(J)
S               CLA;TAD \I      /DIFFERENCE
S               JMS IWRITE
S               CLA;TAD (6      /MESSAGE 6
S               JMS CWRITE;CLA
        IF(J) 63,90,64
63      CONTINUE
S               CLA;TAD (7      /MESSAGE 7
S               JMS CWRITE
S               JMP \90
64      CONTINUE
S               CLA;TAD (D10    /MESSAGE 10
S               JMS CWRITE
S               JMP \90
C       TO 70 FOR COMMAND ADD
70      IADD=0
        DO 72 I=1,7
        J=IP(I)
        IF(J) 73,73,71
71      IADD=IADD+IL(J)
72      CONTINUE
73      CONTINUE
S               CLA;TAD (D9     /MESSAGE 9
S               JMS CWRITE
S               TAD \IADD       /ADDITION
S               JMS IWRITE
S               TAD (6          /MESSAGE 6
S               JMS CWRITE
S               JMP \90
```

```
C       THIS IS THE MAIN COMMAND SUBROUTINE
S       COMAND, 0
C       CLEAR PARAMETER ARRAY
        DO 100 I=1,7
100     IP(I)=0
S               CLA
S               6131    /CLSK: CHECK CLOCK FLAG
S               SKP;JMS CLOFF
S               CLA
C       READ ENTRY TIME AND STORE
        INA=INA+1
        NTIME(INA)=I10
        I10=0
S               CLA;TAD LIST
S               DCA PNTR
S               TAD (D-20
S               DCA COUNT
S       LOOP1,  INC PNTR
S               CLA;DCA I PNTR  /ZERO LIST
S               ISZ COUNT
S               JMP LOOP1
S               TAD (215
S               JMS TYPE
S               TAD (212        /OUTPUT "CR,LF,*"
S               JMS TYPE
S               TAD (252
S               JMS TYPE
S       WAIT1,  6131    /CLSK
S               SKP
S               JMS CLOFF
S               KSF             /WAIT FOR FIRST KEY PUSH
S               JMP WAIT1
S               CLA
C       READ TIME OF FIRST KEY AND STORE
        INA=INA+1
        NTIME(INA)=I10
        I10=0
S       BEGIN,  CLA;TAD LIST
S               DCA PNTR
S       START,  JMS READ        /GET CHARACTER
S               DCA TEMP        /STORE AND CHECK FOR
S               TAD TEMP
S               TAD (-215
S               SNA;JMP RETURN  / "RETURN"
S               CLA;TAD TEMP
S               TAD (-375
S               SNA;JMP ALTMO   / "ALT MODE"
S               CLA;TAD TEMP
S               TAD (-254
S               SNA;JMP OK      / ","
S               CLA;TAD TEMP
S               TAD (-377
S               SNA;JMP RUBOUT  / "RUBOUT"
S               CLA;TAD TEMP
S               TAD (-240
S               SNA;JMP OK      / "SPACE"
S               CLA;TAD TEMP
S               TAD (-261
S               SPA;JMP START   / <"0"
C
```

```
S               CLA;TAD TEMP
S               TAD (-270
S               SPA;JMP OK          / >"7"
S               CLA;TAD TEMP
S               TAD (-301
S               SPA;JMP START       / >"A"
S               CLA;TAD TEMP
S               TAD (-333
S               SMA;JMP START       / <"Z"
S       OK,     CLA
S               INC PNTR
S               TAD PNTR
S               CIA                 /IF LIST FULL
S               TAD LIST
S               TAD (D21
S               SNA;JMP ENOUGH      / JMP TO ENOUGH
S               CLA;TAD TEMP
S               JMS TYPE            /IF NOT FULL 'ECHO' CHAR
S               TAD TEMP
S               DCA I PNTR          /AND STORE IN LIST
S               JMP START
S       ENOUGH, STA                 /TO HERE ON 21ST CHAR
S               TAD PNTR
S               DCA PNTR
S       LOOP2,  JMS READ            /ONCE IN THIS LOOP, ONLY
S               DCA TEMP
S               TAD TEMP
S               TAD (-215
S               SNA;JMP RETURN      / "RETURN" OR
S               CLA;TAD TEMP
S               TAD (-375
S               SNA;JMP ALTMO       / "ALT MODE" OR
S               CLA;TAD TEMP
S               TAD (-377
S               SNA;JMP RUBOUT      / "RUBOUT"
S               JMP LOOP2           / WILL GET YOU OUT
S       RUBOUT, CLA                 /RUBOUT SEQUENCE
S               TAD LIST
S               CIA
S               TAD PNTR
S               SZA                 /ARE YOU RUBBING OUT PAST START
S               JMP RUBIT           /NO: DELETE CHAR
S               TAD (277            /YES: TYPE "?"
S               JMS TYPE
S               JMP START
S       RUBIT,  CLA
S               DCA I PNTR          /ZERO LIST ELEMENT
S               STA
S               TAD PNTR
S               DCA PNTR            /RESET POINTER
S               TAD (243
S               JMS TYPE            / TYPE "#"
S               JMP START
S       PNTR,   0
S       TEMP,   0
```

```
S               CPAGE 25
S       LIST,   LIST
S               0;0;0;0;0;0;0;0;0;0     / INPUT CHARACTERS
S               0;0;0;0;0;0;0;0;0;0     /STORED HERE
S               0;0
S       RETURN, CLA;TAD (215    / WHEN "RETURN" IS PUSHED
S               JMS TYPE        / OUTPUT "CR,LF"
S               TAD (212
S               JMS TYPE
S               TAD LIST
S               DCA PNTR
S               INC PNTR
S               TAD I PNTR      /GET 1ST CHAR
S               AND (0077
S               RTL;RTL;RTL     / PACK IN BITS 0-5 OF WORD1
S               DCA WORD1
S               INC PNTR
S               TAD I PNTR      /GET 2ND CHAR
S               AND (0077       / PACK IN BITS 6-11 OF WORD1
S               TAD WORD1
S               DCA WORD1
S               INC PNTR
S               TAD I PNTR      / GET 3RD CHAR
S               AND (0077
S               RTL;RTL;RTL     / PACK IN BITS 0-5 OF WORD2
S               DCA WORD2
S               INC PNTR
S               TAD I PNTR      /GET 4TH CHAR
S               AND (0077       /PACK IN BITS 6-11 OF WORD2
S               TAD WORD2
S               DCA WORD2
S               TAD (-2
S               DCA SWITCH      /SET UP SWITCHES
S               TAD WORD1
S               DCA WORD
S               STA
S       LOOP3A, TAD PATRN       /SETUP PATTERN LIST POINTERS
S               DCA POINT
S               DCA COUNT
S               STA;DCA COUNT1
S       LOOP3,  INC POINT
S               INC POINT       / CSAN LIST
S               CLA IAC
S               TAD COUNT
S               DCA COUNT
S               TAD I POINT     /COMP WORD WITH PATTERN
S               SNA;JMP CHECK
S               CIA
S               TAD WORD
S               SNA;JMP FOUND   /IF MATCH FOUND GOTO FOUND
S               CLA;JMP LOOP3
S       FOUND,  CLA IAC
S               TAD COUNT1
S               DCA COUNT1      /STORE NUMBER OF PASSES BEFORE
S               TAD COUNT       /MATCH WAS FOUND
S               DCA FIND        /IN FIND. AND NUMBER OF FINDS
S               JMP LOOP3       /IN COUNT1.
```

```
S       CHECK,  ISZ SWITCH
S               JMP PASS1           /CHECK WHICH PASS
S               JMP PASS2
S       PASS1,  CLA;TAD COUNT1      / PASS1
S               SNA;JMP GO          /IF ONLY 1 MATCH JMP GO
S               SPA;JMP NOGO        /IF NO MATCH FOUND JMP NOGO
S               CLA;TAD WORD2       /IF >1 MATCH FOUND CONTINUE
S               DCA WORD
S               JMP LOOP3A
S       PASS2,  CLA;TAD COUNT1      / PASS 2
S               SNA;JMP GO          / IF ONLY 1 MATCH JMP GO
S               SMA;JMP NOGO        /IF NO MATCH JMP NOGO
S               CLA;TAD ("M         /IF >1 MATCHES TYPE"MORE" JMP ALTMO
S               JMS TYPE
S               TAD ("O
S               JMS TYPE
S               TAD ("R
S               JMS TYPE
S               TAD ("E
S               JMS TYPE
S               TAD ("!
S               JMS TYPE
S               JMP ALTMO
S       WORD1,  0
S       WORD2,  0
S       FIND,   0
S       SWITCH, 0
S       POINT,  0
S       WORD,   0
S       COUNT,  0
S               CPAGE 23
S       PATRN,  PATRN               /PATTERN LIST
S               1716;4040           /ON--
S               1706;0640           /OFF-
S               2325;2020           /SUPP
S               2324;0124           /STAT
S               0405;1501           /DEMA
S               2303;1722           /SCOR
S               0104;0440           /ADD-
S               0000;0000           /TERMINATOR
S       NOGO,   CLA                 / TO HERE ON NO MATCH
S               TAD (305
S               JMS TYPE            /TYPE "EH
S               TAD (310
S               JMS TYPE
S       ALTMO,  TAD (277
S               JMS TYPE
S               TAD LIST
S               DCA PNTR
S               TAD (D-20
S               DCA COUNT
S       LOOP4,  INC PNTR
S               CLA
S               DCA I PNTR          /ZERO LIST
S               ISZ COUNT
S               JMP LOOP4
S               JMP BEGIN           / START OVER
```

```
S       STORE,  0
S       COUNT1, 0
S       GO,     CLA             /TO HERE ON UNIQUE MATCH ONLY
S               TAD (D-8        /GET FIRST 8 NUMBERS FROM
S               DCA COUNT1      /LIST INTO PARAMETER ARRAY
S               DCA \I
S               TAD LIST
S               DCA PNTR
S       LOOP5,  CLA
S               INC PNTR
S               TAD I PNTR      /GET CHAR
S               SNA;JMP OUT
S               TAD (-260       /IS IT >"0"
S               SPA;JMP LOOP5   /YES: LOOP
S               DCA STORE       /NO: STORE
S               TAD STORE
S               TAD (D-8        /IS IT >8
S               SMA;JMP LOOP5   /YES: LOOP
S               ISZ COUNT1      /NO: IS IT 8TH CHAR
S               SKP;JMP OUT     /YES: JMP OUT
S               CLA IAC         /NO: INC POINTER I
S               TAD \I
S               DCA \I
S               TAD STORE
S               DCA \J
C       STORE IT IN IP(I)
        IP(I)=J
S               JMP LOOP5
S       OUT,    CLA
S               6131    /CLSK: CHECK CLOCK FLAG
S               SKP;JMS CLOFF
C       READ EXIT TIME AND STORE
        INA=INA+1
        NTIME(INA)=I10
        I10=0
C       IF THERE ARE NO PARAMETERS LISTES SET IP TO 1-7 INCLUSIVE
        J=IP(1)
S               CLA;TAD \J
S               SZA;JMP EXIT    /IF IP(1) NE 0 JMP EXIT
S               CLA
        DO 101  I=1,7
101     IP(I)=I
S       EXIT,   CLA
S               DCA DATA
        DO 111 I=1,7
        J=IP(I)
        IF(J) 112,112,110
110     CONTINUE
S               CLA;TAD \J
S               CIA;DCA SHIFT
S               TAD (1
S       LOOP11, ISZ SHIFT
S               SKP;JMP OUT10
S               RAL
S               JMP LOOP11
S       OUT10,  TAD DATA
S               DCA DATA
111     CONTINUE
112     CONTINUE
```

```
S               CLL CLA;TAD FIND
S               RTL;RTL;RTL;RAL
S               TAD DATA
S               DCA DATA
S               CLL CLA;TAD \IE
S               RAR;RTR
S               TAD DATA
S               DCA \J
        IND=IND+1
        ICOM(IND)=J
S               CLA
S               TAD FIND        /PUT IN ACC NUMBER OF COMMAND
S               JMP I COMAND
S       DATA,   0
S       SHIFT,  0
C
```

```
S       TYPE,   0               /TYPE ROUTINE
S               DCA LOCAL       /STORE CHAR LOCALLY
S       WAIT2,          6131    /CLSK: CHECK CLOCK FLAG
S               SKP
S               JMS CLOFF
S               TSF
S               JMP WAIT2       /WAIT FOR FLAG
S               TAD LOCAL       /GET CHAR
S               TLS             /TYPE IT
S               CLL CLA IAC
S               TAD \IC
S               DCA \IC /INC CHAR COUNT
S               JMP I TYPE
C
S       READ,   0               /READ ROUTINE
S       WAIT3,  6131    /CLSK: CHECK CLOCK FLAG
S               SKP
S               JMS CLOFF
S               KSF
S               JMP WAIT3       /WAIT FOR FLAG
S               CLA CLL
S               KRB             /READ CHAR
S               JMP I READ
S       LOCAL,  0
```

```
S       CLOFF,  0               / TO HERE ON CLOCK OFLOW
S               6135            /CLSA: CLEAR STATUS
S               CLA
C       INCREMENT 1/10 TH SEC COUNTER I10
        I10=I10+1
S               ISZ COUNTR      /INC 10 SEC COUNTER IF 10 SECS UP SKP
S               JMP I CLOFF
S               CLA;TAD (-144   / RESET COUNTER
S               DCA COUNTR
C       ADD UP VALUE OF THOSE AREAS ON
        IT=0
        DO 102 I1=1,7
        J1=IS(I1)
S               CLA;TAD \J1
S               SNA;JMP \102
S               CLA
        IT=IT+IL(I1)
102     CONTINUE
C       STORE TOTAL OF THOSE ON
        INC=INC+1
        ITOT(INC)=IT
S               CLA
S               ISZ COUNT7
S               SKP             / EVERY 6 JMP CHANGE
S               JMS CHANGE
S               JMP I CLOFF
S       COUNT7, -6
S       COUNTR, -144
```

```
S       CHANGE, 0               / CHANGE ROUTINE
S               CLA;TAD (-6
S               DCA COUNT7
        INB=INB+1
S               CLA;TAD \INB
S               CIA
S               TAD \ITIME      /IF TIME UP EXIT
S               SNA;JMS FINISH
S               CLA
C       IMPLEMENT CHANGE
        I1=IRAND(1,INB)
        IF(I1) 104,104,105
104     ISUP=IRAND(2,INB)
        GOTO 106
105     IL(I1)=IRAND(2,INB)
106     CONTINUE
S               CLA;JMS BELLS
S               JMP I CHANGE
```

```
S       BELLS,  0               /TO RING BELL
S               CLA;TAD (207
S       WAIT5,  TSF
S               JMP WAIT5
S               TLS;CLA
S               JMP I BELLS
```

```
S       FINISH, 0               /EXIT ROUTINE WRITES DATA TO TAPE
S               CLA
        WRITE(1,200)
200     FORMAT(/,'YOU HAVE FINISHED. THANK YOU VERY MUCH, GOODBYE!',/)
        I=2
        WRITE(4,400) ICOM,NTIME,ITOT,I,ISUB,IGRP,IRUN,X1
400     FORMAT(1254A2,A6)
        CALL OCLOSE
        CALL EXIT
```

```
C       THIS PART OF THE MAIN PROGRAM WRITES MESSAGES TO THE TTY.
C       THE NUMBER OF THE MESSAGE TO BE WRITTEN IS IN THE ACC ON ENTRY
C       IT CAN BE INTERRUPTED, PUSHING A KEY WILL SET IE NE 0
C
S       CWRITE, 0
S               DCA \N          /STORE MESSAGE NUMBER
        KA=0
1 100   KA=KA+1
        NA=MESS(KA,N)
S               CLA;TAD \NA     /GET PACKED WORD FROM ARRAY
S               AND (7700
S               RTR;RTR;RTR     /GET LEFT HALF
S               JMS TEST
S               TAD \NA
S               AND (0077       /GET RIGHT HALF
S               JMS TEST
S               JMP \1100       /JMP ARROUND
S       TEST,   0
S               SNA;JMP OUT1    /IF 00 EXIT
S               DCA STORE1
S               TAD \IE         /IF IE NE 0 EXIT
S               SZA;JMP OUT1
S               TAD STORE1
S               TAD (-37
S               SNA;JMP CRLF    /IF CHAR IS "-" JMP CRLF
S               CLA CLL;TAD STORE1
S               RTL;RTL;RTL;RAL /PUT BIT 7 IN LINC
S               SZL;JMP L200    / IF LINC = 1 SKIP
S               CLA CLL;TAD STORE1
S               TAD (300        /ADD 300
S               JMS TIPE        /TYPE CHAR
S               JMP I TEST
S       L200,   CLA CLL;TAD STORE1
S               TAD (200        /ADD 200
S               JMS TIPE        /TYPE CHAR
S               JMP I TEST
S       CRLF,   TAD (215        / TYPE "CR,LF"
S               JMS TIPE
S               TAD (212
S               JMS TIPE
S               JMP I TEST
S       STORE1, 0
S       OUT1,   CLA
S               JMP I CWRITE
```

```
C       THIS PART OF THE MAIN PROGRAM WRITES UNSIGNED INTEGERS
C       ARRIVING IN OCTAL, IN DECCIMAL ON THE TTY. INTEGERIS IN
C       ACCUMULATOR ON ENTRY
C
S       PAGE
S       IWRITE, 0
S               DCA VALUE           /STORE VALUE
S               DCA DIGIT           /CLEAR DECIMAL DIGIT COUNTER
S               DCA SW1
S               TAD COUNT2
S               DCA COUNT3          /SET UP SWITCHES AND COUNTERS
S               TAD COUNT2
S               DCA COUNT4
S               TAD ADDR
S               DCA ARROW
S               SKP
S       LOOP6,  DCA VALUE
S               CLL
S       LOOP7,  TAD VALUE           /GET OCTAL NUMBER
S       ARROW,  TAD TENPWR          /SUB DECIMAL
S               SZL
S               INC DIGIT           /UNTIL -VE RESULTS
S               SZL                 /DIGIT HOLDS NUMBER OF SUBTRACTIONS
S               JMP LOOP6
S               CLA;TAD \IE         /IF IE NE 0 THEN EXIT
S               SZA;JMP XOUT
S               TAD DIGIT           /IF DIGIT IS 0
S               SZA;DCA SW1         /LEAVE SW1 = 0
S               TAD SW1             /IF SW1 = 0
S               SNA;JMP EXIT2       /JMP EXIT2
S               CLA                 /IF NOT
S       IN1,    TAD DIGIT           /GET DIGIT
S               TAD (260            /ADD 260. CONVERT TO ASCII
S               JMS TIPE            /TYPE IT
S       OUT2,   DCA DIGIT           /CLEAR IT
S               ISZ ARROW           /**GET NEXT SUBTRACTAND**
S               ISZ COUNT3          /CHECK FOR ENOUGH
S               JMP LOOP7
S               JMP XOUT
S       EXIT2,  CLA;TAD (240        / ADD "SPACE"
S               JMS TIPE            /TYPE IT
S               ISZ COUNT4          /CHECK FOR 3 .ON 4TH 0 TYPE 0
S               JMP OUT2
S               CLA;DCA DIGIT
S               JMP IN1
S       XOUT,   CLA
S               JMP I IWRITE
S       ADDR,   TAD TENPWR
S       COUNT2, -4
S               CPAGE 4
S       TENPWR, -1750    /OCTAL 1000
S               -144     /OCTAL 100
S               -12      /OCTAL 10
S               -1       /OCTAL 1
S       VALUE,  0
S       DIGIT,  0
S       COUNT3, 0
S       COUNT4, 0
S       SW1,    0
```

```
C       TYPE ROUTINE FOR MESSAGE OUTPUT
S       TIPE,   0
S               DCA LOCAL1          /STORE WHAT IS TO BE OUTPUT
S       WAIT4,  6131    /CLSK       CHECK CLOCKFLAG
S               SKP
S               JMS CLOFF
S               TSF
S               JMP WAIT4           / WAIT FOR FLAGS
S               KSF;SKP             /HAS KEY BEEN STRUCK ?
S               JMP INRPT           / YES: JMP INRPT
S               CLA;TAD LOCAL1      /NØ: GET CHAR
S               TLS                 /TYPE IT
S               CLL CLA IAC
S               TAD \IC             /INC CHAR COUNTER
S               DCA \IC
S               JMP I TIPE
S       INRPT,  KRB                 /READ INTERRUPTING CHAR
S               CLA IAC
S               DCA \IE             /SET IE NE Ø
S               CLL;JMP I TIPE
S       LOCAL1, Ø
        END
```

```
C
C       START
C
C       R.G.ASHTON      HUSAT   LOUGHBOROUGH UNIVERSITY         1972
C
C       THIS IS THE INITILIZATION SUBROUTINE FOR THE "IRATE" TASKS
C       IT GETS ALL THE INCOMMING FILES DATA AND OPENS THE OUTPUT FILE.
C
        SUBROUTINE START
        COMMON I10,ISUP,IP,IS,IL,MESS,ITIME,IRAND,ISUB,IGRP,IRUN,X1
        DIMENSION IP(7),IL(7),IS(7),MESS(10,10),INIT(8),IRAND(2,29)
        WRITE(1,200)
        READ(1,100) X
        CALL IOPEN ('DTA1',X)
        READ(4,400) MESS
        WRITE(1,201)
        READ(1,100) X1
        CALL IOPEN ('DTA1',X1)
        READ(4,400) INIT,IRAND
        WRITE(1,202)
        READ(1,100) X
        CALL OOPEN('DTA1',X)
        WRITE(1,205)
        READ(1,103) ISUB,IGRP,IRUN
        WRITE(1,203)
        READ (1,101) ITIME
        DO 10 I=1,7
        IS(I)=0
10      IL(I)=INIT(I)
        ISUP=INIT(8)
        WRITE(1,204)
S               HLT;NOP
S               CLA
S               6134    /CLEN
S               6133    /CLAB
S               6132    /CLLR
S               TAD (100
S               6132    /CLLR
S               6135    /CLSA
S               CLA
S               TAD (D1000
S               CIA
S               6133    /CLAB
S               CLA
S               TAD (300
S               6134    /CLEN
S               CLA
S               TAD (3100
S               6132    /CLLR
S               CLA
        RETURN
100     FORMAT(A6)
101     FORMAT(I2)
102     FORMAT(I4)
103     FORMAT('SUBJECT NO = ',I2,/,'GROUP NO    = ',I1,/,'TRIAL NO    = '
        1,I1,/)
200     FORMAT(/,'MESS SET FILE NAME, A6',/)
201     FORMAT(/,'CHANGE SET FILE NAME, A6',/)
202     FORMAT(/,'DATA SET FILE NAME, A6',/)
203     FORMAT(,/'NUMBER OF CHANGES',/)
204     FORMAT(/,'PUSH CONTINUE',/)
205     FORMAT(/,'ENTER SUBJECTS DATA')
400     FORMAT(A2)
        END
```

```
C
C       WTAPE
C
C       R.G.ASHTON      HUSAT   LOUGHBOROUGH UNIVERSITY         1972
C
C       THIS PROGRAM WILL ACCEPT UPTO 10 MESSAGES EACH OF UPTO 19
C       CHARACTERS AND WRITE THEM TO A SPECIFIED FILE ON A SPECIFIED
C       DEVICE. THE DEVICE AND FILE NAMES ARE ENTERED DURING THE
C       RUNNING OF THE PROGRAM, THEY MUST CONTAIN EXACTLY 6 CHARACTERS
C       MADE UP IF NECESSARY WITH "0". THE PROGRAM WILL EXIT IF
C       MORE THAN 10 MESSAGES ARE INPUT OR IF AN "ALT MODE" IS USED
C       TO TERMINATE A MESSAGE. "@" MUST NOT BE USED IN THE MESSAGE.
C       THE CHARACTERS ARE STORED IN STRIPPED ASCII FORM
C
C       THE PROGRAM USES NO PREFISX AND NORMAL LABELS SO CARE MUST
C       BE TAKEN IF ANOTHER PROGRAM IS APPENDED.
C
        DIMENSION M(10,10),IDUM(20)
        DO 1000 I=1,10
        DO 1000 J=1,10
1000    M(I,J)=0
        WRITE(1,200)
200     FORMAT(/,'ENTER DEVICE AND FILE NAMES IN A6 FORMAT',/)
        READ(1,100)X1
        READ(1,100)X2
100     FORMAT(A6)
        WRITE(1,201)
201     FORMAT(/,'YOU MAY INPUT UP TO 10 MESSAGES, EACH OF UP TO 19
        9CHARACTERS',/,'TERMINATE THE LAST MESSAGE WITH ALT MODE',/)
        K=0
2       K=K+1
        IF(K-10) 3,3,4
3       DO 1001 I=1,20
1001    IDUM(I)=0
S
S                       JMS STORE       /JUMP TO STORE ROUTINE
S                       JMP \2          /EXITS TO HERE ON "RETURN"
S                       JMP \5          /EXITS TO HERE ON "ALTMODE"
S
4       WRITE(1,202)
202     FORMAT(/,'YOU HAVE INPUT 10 MESSAGES',/)
5       CALL OOPEN(X1,X2)
        WRITE(4,400) M
400     FORMAT(A2)
        CALL OCLOSE
        CALL EXIT
```

```
S
S               STORE,  0
S                       CLA;CLL
S
        I=0
S               START,  JMS READ         /GET CHARACTER
S                       DCA TEMP         /STORE IN TEMP
S                       TAD TEMP
S                       JMS TYPE         /TYPE IT
S                       TAD TEMP
S                       TAD     (-215
S                       SNA              /IS IT "RETURN" ?
S                       JMP PACK         /YES: JUMP
S                       CLA;CLL /NO: CLEAR
S                       TAD TEMP
S                       TAD     (-375
S                       SNA              /IS IT "ALTMODE" ?
S                       JMP ALTMO        /YES: JUMP
S                       CLA;CLL /NO: CLEAR
S                       TAD TEMP
S                       TAD     (-377
S                       SNA              /IS IT "RUB OUT" ?
S                       JMP RUBOUT       /YES: JUMP
S                       CLA;CLL          /NO: CLEAR
        I=I+1
        IF(I-19) 6,6,9
6       CONTINUE
S                       CLA;CLL
S                       TAD TEMP
S                       DCA \J           /STORE CHAR IN J
        IDUM(I)=J
S                       JMP START        /GET NEXT CHAR
S               RUBOUT, CLA;CLL
        IF(I-1) 7,8,8
7       CONTINUE
S                       TAD     (277
S                       JMS TYPE         /TYPE "?"
S                       TAD     (207
S                       JMS TYPE         /TYPE "BELL"
S                       JMP START
8       CONTINUE
S                       CLA;CLL
        IDUM(I)=0
        I=I-1
S                       TAD     (243
S                       JMS TYPE         /TYPE "#"
S                       JMP START
S
S               ALTMO,  INC STORE        /INC RETURN ADRESS
9       CONTINUE
```

```
S               PACK,   CLA;CLL /PACK UP ARRAY TO 36 CHARS
S                       TAD     (215
S                       JMS TYPE        /TYPE "RETURN"
S                       TAD     (212
S                       JMS TYPE        /TYPE "LINE FEED"
        I=0
        J=0
10      I=I+1
        IF(I-20)11,11,12
11      J=J+1
        IX=IDUM(I)
S                       CLA;CLL
S                       TAD \IX         /PUT CHAR IN ACC
S                       AND     (0077   /MASK BITS 6-11
S                       RTL;RTL;RTL     /ROTATE RIGHT 6 BITS
S                       DCA \IY         /STORE IN \IY
        I=I+1
        IX=IDUM(I)
S                       CLA;CLL
S                       TAD \IX         /PUT NEXT CHAR IN ACC
S                       AND     (0077   /MASK BITS 6-11
S                       TAD \IY         /GET PREVIOUS STRIPPED CHAR
S                       DCA \IY         /STORE BOTH IN \IY
        M(J,K)=IY
S                       CLA;CLL
S                       JMP \10         /GET NEXT CHAR TO PACK
12      CONTINUE
S                       JMP I STORE
S
S               READ,   0               /USUAL GET CHAR ROUTINE
S                       CLA;CLL
S               WAIT1,  KSF
S                       JMP WAIT1
S                       KRB
S                       JMP I READ
S
S               TYPE,   0               /USUAL TYPE CHAR ROUTINE
S               WAIT2,  TSF
S                       JMP WAIT2
S                       TLS
S                       CLA;CLL
S                       JMP I TYPE
S
S               TEMP,   0               /TEMPORARY STORAGE
        END
```

```
C
C       R.G.ASHTON      HUSAT   LOUGHBOROUGH UNIVERSITY         1972
C
C       THIS PROGRAM CALLED "WRAND" IS DESIGNED TO SET UP THE FILES
C       REQUIRED BY THE "SUPPLY ALLOCATION TASK".
C       IT READS THE INITIAL VALUES OF THE 7 AREAS USED IN THE TASK
C       AND THE INITIAL SUPPLY. IT ALSO READS THE NEXT 29 VALUES
C       VARIOUS AREAS WILL CHANGE TO AS THE TASK PROGRESSES.
C       AREA 0 IS THE SUPPLY
C
        DIMENSION INIT(8),IRAND(2,29)
10      WRITE(1,200)
        DO 1000 I=1,7
        WRITE(1,201) I,ID,
        READ (1,100) J
        INIT(I)=J
1000    CONTINUE
        WRITE(1,202) ID,
        READ(1,101) J
        INIT(8)=J
        WRITE(1,203)
        DO 1001 I=1,29
        WRITE(1,204) I,ID,
        READ (1,102) J,K
        IRAND(1,I)=J
        IRAND(2,I)=K
1001    CONTINUE
        WRITE(1,206)
S               HLT;NOP
        IF(IRDSW(0)) 10,11,10
11      WRITE(1,207)
        READ(1,103) X
        CALL OOPEN('SYS',X)
        WRITE(4,400) INIT,IRAND
        CALL OCLOSE
        CALL EXIT
100     FORMAT(I2)
101     FORMAT(I3)
102     FORMAT('/   AREA ',I1,' VALUE = ',I3)
103     FORMAT(A6)
200     FORMAT(/,'ENTER THE INITIAL LOADS FOR THE 7 AREAS',/)
201     FORMAT('AREA ',I2,' = ',I0)
202     FORMAT(/,'ENTER THE INITIAL SUPPLY VALUE ',I0)
203     FORMAT(/,'ENTER THE AREA AND VALUE OF 29 CHANGES',/)
204     FORMAT(I2,I0)
206     FORMAT(/,'IF GO SET SW EQ 0,IF NOGO SET SW NE 0, CONTINUE',/)
207     FORMAT(/,'ENTER NAME OF FILE FOR OUTPUT',/)
400     FORMAT(A2)
        END
```

ANNEX B

EXPERIMENTAL INSTRUCTIONS

REAL TIME DISTRIBUTION TASK

INSTRUCTIONS TO SUBJECTS FOR 1st TRIAL

Please read these instructions very carefully

For the purpose of this experiment, you are in the position of an electricity distribution controller. You have one supply and seven areas all demanding some quantity of electricity from the supply. Your task is to supply as much of the total demand as possible by connecting or disconnecting some of the seven areas.

An area can be ON or connected to the supply, in which case it will draw ALL its demand from the supply, or it can be OFF or disconnected from the supply in which case it will draw nothing from the supply. There are no intermediate states between OFF and ON.

Your task is to allocate the supply to various areas, so that the total demand does not exceed the supply. On the other hand you must have as little supply spare as possible. You are required to perform this task for 30 minutes, at frequent intervals during this time the elements of the situation will change. That is the supply or the demand from one of the areas will change in value either up or down. Only ONE element will change in value at a time. Your are required to detect this change and alter your allocation to meet it, if necessary. The computer will not tell you what has changed or when it changed.

There are no minimum or maximum supply requirements for any areas. You can leave any one area on or off all the time if you wish. Your error score, that is the difference between the total demand of those areas ON and the supply, is continuously recorded so that it is unadvisable to turn all the areas ON or OFF as this will affect your error score to a large extent.

In the later analysis your performance will be assessed by calculating the number of possible allocations you could have made that would have been better than the one you chose. Errors of OVERLOADING are counted TWICE as bad as errors of underloading.

COMMAND DECODER

You are required to manipulate the system by using the commands explained below. When you have finished entering any command press 'RETURN'. This commits the computer to execute your command.

The command words are ON, OFF, STATUS, DEMAND, SUPPLY, SCORE and ADD. Any of these commands may be abbreviated to the first two characters, i.e. ON, OF, ST, DE, SU, SC, and AD.

The effect of these commands is as follows:

ON — will connect all areas to the supply.

OFF — will disconnect all areas to the supply.

STATUS — will cause the computer to list the status (ON or OFF) of all areas.

DEMAND — will cause the computer to list the demands of all areas.

SUPPLY — will cause the computer to list the value of the supply.

SCORE — will cause the computer to add up the demand of all those areas whose status is ON and subtract it from the supply and output the difference along with OVERLOADED or UNDERLOADED or nothing as necessary.

ADD — this is a special command which is to help you with the arithmetic. Explained more fully later.

The first four commands above, ON, OFF, STATUS and DEMAND can have their action modified. Using the command on its own terminated by a 'RETURN' will cause the commands to be executed for all areas. However, if instead of a 'RETURN' after the command word a list of numbers is added, and this list is terminated by 'RETURN' then the command will only be executed for the areas numbered in the list.

In the following examples "-" means 'SPACE' and ↵ means 'RETURN'.

Except between the letters of the command word, spaces and/or commas may be typed as you wish.

ON-1,2,3↵ will connect areas 1,2 and 3 to the supply i.e. the status of areas 1,2 and 3 will now be ON.

OF123↵ will disconnect areas 1,2 and 3 to the supply i.e. the status of areas 1,2 and 3 will now be OFF.

STATUS--12,7↵ will list the status of areas 1,2 and 7.

DE4↵ will list the demand of area 4.

The commands below are not used with lists as they are only single item variables.

SU↵ will list the value of the supply.

SCORE↵ will cause the computer to add up the demand of all those areas which have a status ON and subtract this from supply and output the difference along with 'OVERLOADED' or 'UNDERLOADED' or nothing as necessary.

The command ADD is to assist you with the task arithmetic typing.

ADD-123↵ will cause the computer to add the demands of areas 1,2 and 3 and output this value. Regardless of the status of these areas.

Any of the commands that cause the computer to list values such as STATUS, DEMAND, SUPPLY, TOTAL or ADD can be interrupted at any time whilst it is typing by pushing <u>any</u> key. And control will return to you.

When the computer is ready for command it will type a "*" you may then enter your command. If you make a mistake, pushing 'RUBOUT' will delete the previous character and echo a "←". Pushing 'RUBOUT' four times will delete the previous four characters. If you wish, you can delete all you have entered on a line by pushing 'ALTMODE' which will echo a "?".

If the computer cannot understand a command word it will output "EH?" and you must enter the command again. You cannot delete more characters than you have entered, an attempt to do so will cause the computer to output "?".

The 'RETURN' that is used to terminate a command cannot be deleted. If after you have pushed 'RETURN' the computer cannot make sense of your command, it will type "EH?" and you must enter the complete line again.

You will start off with all areas OFF i.e. MAXIMUM UNDERLOADED.

REAL TIME DISTRIBUTION TASK

INSTRUCTIONS TO SUBJECTS FOR 2nd TRIAL

Please read these instructions very carefully

For the purpose of this experiment, you are in the position of an electricity distribution controller. You have one supply and seven areas all demanding some quantity of elctricity from the supply. Your task is to supply as much of the total demand as possible by connecting or disconnecting some of the seven areas.

An area can be ON or connected to the supply, in which case it will draw ALL its demand from the supply, or it can be OFF or disconnected from the supply in which case it will draw nothing from the supply. There are no intermediate states between OFF and ON.

Your task is to allocate the supply to various areas, so that the total demand does not exceed the supply. On the other hand you must have as little supply spare as possible. You are required to perform this task for 30 minutes, at frequent intervals during this time the elements of the situation will change. That is the supply or the demand from one of the areas will change in value either up or down. Only ONE element will change in value at a time. Your are required to detect this change and alter your allocation to meet it, if necessary. The computer will not tell you what has changed but it will ring the bell once when a change occurs.

There are no minimum or maximum supply requirements for any areas. You can leave any one area on or off all the time if you wish. Your error score, that is the difference between the total demand of those areas ON and the supply, is continuously recorded so that it is unadvisable to turn all the areas ON or OFF as this will affect your error score to a large extent.

In the later analysis your performance will be assessed by calculating the number of possible allocations you could have made that would have been better than the one you chose. Errors of OVERLOADING are counted TWICE as bad as errors of underloading.

COMMAND DECODER

You are required to manipulate the system by using the commands explained below. When you have finished entering any command press 'RETURN'. This commits the computer to execute your command.

The command words are ON, OFF, STATUS, DEMAND, SUPPLY, SCORE and ADD. Any of these commands may be abbreviated to the first two characters, i.e. ON, OF, ST, DE, SU, SC, and AD.

The effect of these commands is as follows:

ON — will connect all areas to the supply.

OFF — will disconnect all areas to the supply.

STATUS — will cause the computer to list the status (ON or OFF) of all areas.

DEMAND — will cause the computer to list the demands of all areas.

SUPPLY — will cause the computer to list the value of the supply.

SCORE — will cause the computer to add up the demand of all those areas whose status is ON and subtract it from the supply and output the difference along with OVERLOADED or UNDERLOADED or nothing as necessary.

ADD — this is a special command which is to help you with the arithmetic. Explained more fully later.

The first four commands above, ON, OFF, STATUS and DEMAND can have their action modified. Using the command on its own terminated by a 'RETURN' will cause the commands to be executed for all areas. However, if instead of a 'RETURN' after the command word a list of numbers is added, and this list is terminated by 'RETURN' then the command will only be executed for the areas numbered in the list.

In the following examples "-" means 'SPACE' and ↵ means 'RETURN'.

Except between the letters of the command word, spaces and/or commas may be typed as you wish.

ON-1,2,3↵ will connect areas 1,2 and 3 to the supply i.e. the status of areas 1,2 and 3 will now be ON.

OF123↵ will disconnect areas 1,2 and 3 to the supply i.e. the status of areas 1,2 and 3 will now be OFF.

STATUS--12,7↵ will list the status of areas 1,2 and 7.

DE4↵ will list the demand of area 4.

The commands below are not used with lists as they are only single item variables.

SU↵ will list the value of the supply.

SCORE↵ will cause the computer to add up the demand of all those areas which have a status ON and subtract this from supply and output the difference along with 'OVERLOADED' or 'UNDERLOADED' or nothing as necessary.

The command ADD is to assist you with the task arithmetic typing.

ADD-123↵ will cause the computer to add the demands of areas 1,2 and 3 and output this value. Regardless of the status of these areas.

Any of the commands that cause the computer to list values such as STATUS, DEMAND, SUPPLY, TOTAL or ADD can be interrupted at any time whilst it is typing by pushing <u>any</u> key. And control will return to you.

When the computer is ready for command it will type a "*" you may then enter your command. If you make a mistake, pushing 'RUBOUT' will delete the previous character and echo a "←". Pushing 'RUBOUT' four times will delete the previous four characters. If you wish, you can delete all you have entered on a line by pushing 'ALTMODE' which will echo a "?".

If the computer cannot understand a command word it will output "EH?" and you must enter the command again. You cannot delete more characters than you have entered, an attempt to do so will cause the computer to output "?".

The 'RETURN' that is used to terminate a command cannot be deleted. If after you have pushed 'RETURN' the computer cannot make sense of your command, it will type "EH?" and you must enter the complete line again.

You will start off with <u>all</u> areas OFF i.e. MAXIMUM UNDERLOADED.

OL/GRP4

ON-LINE DISTRIBUTION TASK

INSTRUCTIONS TO SUBJECTS FOR 1st TRIAL

Please read these instructions very carefully

For the purpose of this experiment, you are in the position of an electricity distribution controller. You have one supply and seven areas all demanding some quantity of electricity from the supply. Your task is to supply as much of the total demand as possible by connecting or disconnecting some of the seven areas.

An area can be ON or connected to the supply, in which case it will draw ALL its demand from the supply, or it can be OFF or disconnected from the supply in which case it will draw nothing from the supply. There are no intermediate states between OFF and ON.

Your task is to allocate the supply to various areas, so that the total demand does not exceed the supply. On the other hand you must have as little supply spare as possible. You are required to perform this task for 30 periods representing $\frac{1}{2}$ hours in real time. When you are satisfied with your allocation for a period, you tell the computer. It then calculates your score and outputs it, and changes one element of the situation. Either a load or the supply will change either up or down. You are required to detect this change and re-allocate your supply if necessary to accomodate the change. Only you can change the status of the areas. You have 30 minutes to complete the task. Your error score will be increased substantially for every minute you use over 30. There are no minimum or maximum supply requirements for any areas. You can leave any one area off or on all the time if you wish. Your error score is the difference between the total demand of those areas ON and the supply.

In the later analysis your performance will be assessed by calculating the number of possible allocations you could have made that would have been better than the one you chose, if there are any. Errors of OVERLOADING are counted TWICE as bad as errors of underloading.

COMMAND DECODER

You are required to manipulate the system by using the commands explained below. When you have finished entering any command press 'RETURN'. This commits the computer to execute your command.

The command words are ON, OFF, STATUS, DEMAND, SUPPLY SCORE, and ADD. Any of these commands may be abbreviated to the first two characters, i.e. ON, OF, DE, SU, SC, and AD.

The effect of these commands is as follows:

ON	will connect <u>all</u> areas to the supply.
OFF	will disconnect <u>all</u> areas to the supply.
STATUS	will cause the computer to list the status (ON or OFF) of all areas.
DEMAND	will cause the computer to list the demands of <u>all</u> areas.
SUPPLY	will cause the computer to list the value of the supply.
SCORE	You type this only when you are happy with the allocation as it terminates one period. The computer gives you a SCORE which is the difference between the total of those areas whose status is on and the supply. It adds UNDERLOADED, OVERLOADED or nothing as necessary. It then CHANGES one element of the situation and returns control to YOU.
ADD	this is a special command explained full later.

The first four commands above, ON, OFF, STATUS and DEMAND can have their action modified. Using the command on its own terminated by a 'RETURN' will cause the commands to be executed for <u>all</u> areas. However, if instead of a 'RETURN' after the command word a list of numbers is added and this list is terminated by 'RETURN' then the command will only be executed for the areas numbered in the list.

In the following examples "-" means 'SPACE' and ↵ means 'RETURN'.

Except between the letters of the command word, spaces and/or commas may be typed as you wish.

ON-1,2,3↵	will connect areas 1, 2 and 3 to the supply i.e. the status of areas 1, 2 and 3 will now be ON.
OF123↵	will disconnect areas 1, 2 and 3 to the supply i.e. the status of areas 1, 2 and 3 will now be OFF.

STATUS--12,7↵ will list the status of areas 1, 2 and 7.
DE4↵ will list the demand of area 4.

The commands below are not used with lists as they are only single item variables.

SU↵ will list the value of the supply.
SCORE↵ You type this only when you are happy with the allocation as it terminates one period. The computer gives you a SCORE which is the difference between the total of those areas whose status is on and the supply. It adds UNDERLOADED, OVERLOADED or nothin as necessary. It then CHANGES one element of the situation and returns control to YOU.

The command ADD is to assist you with the task arithmetic typing.

ADD-123↵ will cause the computer to add the demands of areas 1, 2 and 3 and output this value. Regardless of the status of these areas.

Any of the commands that cause the computer to list values such as STATUS, DEMAND, SUPPLY, SCORE or ADD can be interrupted at any time whilst it is typing by pushing <u>any</u> key. And control will return to you.

When the computer is ready for command it will type a "*" you may then enter your command. If you make a mistake, pushing 'RUBOUT' will delete the previous character and echo a "\". Pushing 'RUBOUT' four times will delete the previous four characters. If you wish you can delete all you have entered on a line by pushing 'ALTMODE' which will echo a "?".

If the computer cannot understand a command word it will output "EH?" and you must enter the command again. You cannot delete more characters than you have entered, an attempt to do so will cause the computer to output "?".

The 'RETURN' that is used to terminate a command cannot be deleted. If after you have pushed 'RETURN' the computer cannot make sense of your command, it will type "EH?" and you must enter the complete line again.

You will start off with <u>all</u> areas OFF i.e. MAXIMUM UNDERLOADED.

REAL TIME DISTRIBUTION TASK

INSTRUCTIONS TO SUBJECTS FOR 2nd TRIAL

For the second trial the task and system are identical to the first, except that the Teletype bell will ring once at each change.

REAL TIME DISTRIBUTION TASK

INSTRUCTIONS TO SUBJECTS FOR 2nd TRIAL

For the second trial the task and systems are identical to the first, except that the Teltype bell will not ring at changes.

ON-LINE DISTRIBUTION TASK

INSTRUCTIONS TO SUBJECTS FOR 2nd TRIAL

This task is very similar to the previous one except that the system will change EVERY time you type SCORE and at NO other time. You are required to allocate the supply as before and when you are satisfied to type SCORE which will output your error score and change one element of the situation.

You are required to perform this task for 30 periods representing $\frac{1}{2}$ hour in real time. Your have 30 minutes to complete the allocation for all 30 periods. Your score will be increased substantially for every minute you use over 30.

REAL TIME DISTRIBUTION TASK

INSTRUCTIONS TO SUBJECTS FOR 2nd TRIAL

This task is very similar to the previous one except that the changes will occur at regular intervals, independent of the SCORE command. Only one element of the situation will change at once and you will not be told when a change occurs. You are required to achieve the same objectives as previously and the task will last for 30 minutes.

ANNEX C

ABSOLUTE CUMULATIVE SUM ERROR SCORES AT 10 SECOND INTERVALS

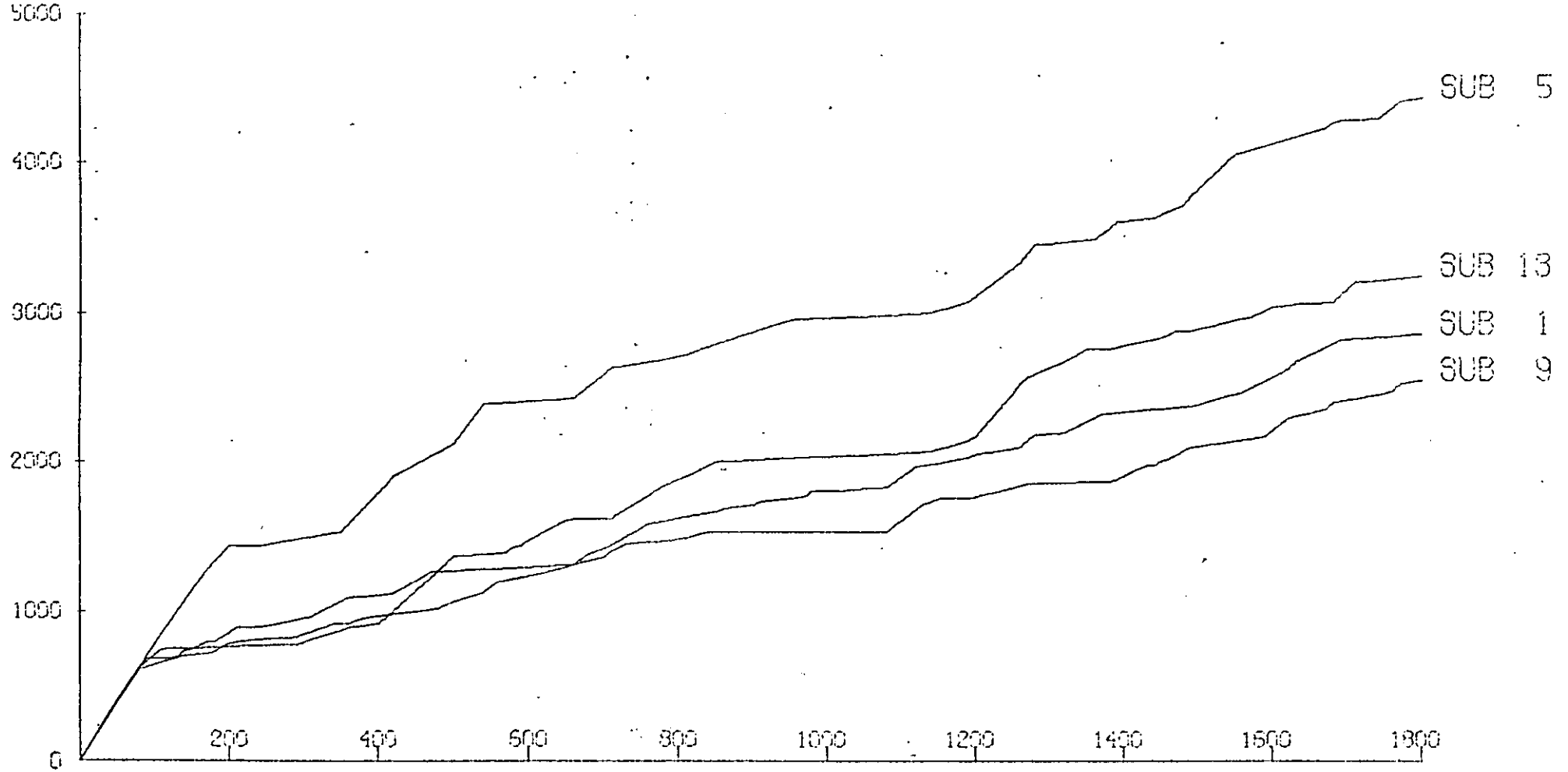

5000
4000
3000
2000
1000
0
200
400
600
800
1000
1200
1400
1600
1800
SECS
SUB 5
SUB 13
SUB 1
SUB 9
ABSOLUTE CUSUM FOR GROUP 1 ON TRIAL 1

Fig 1C

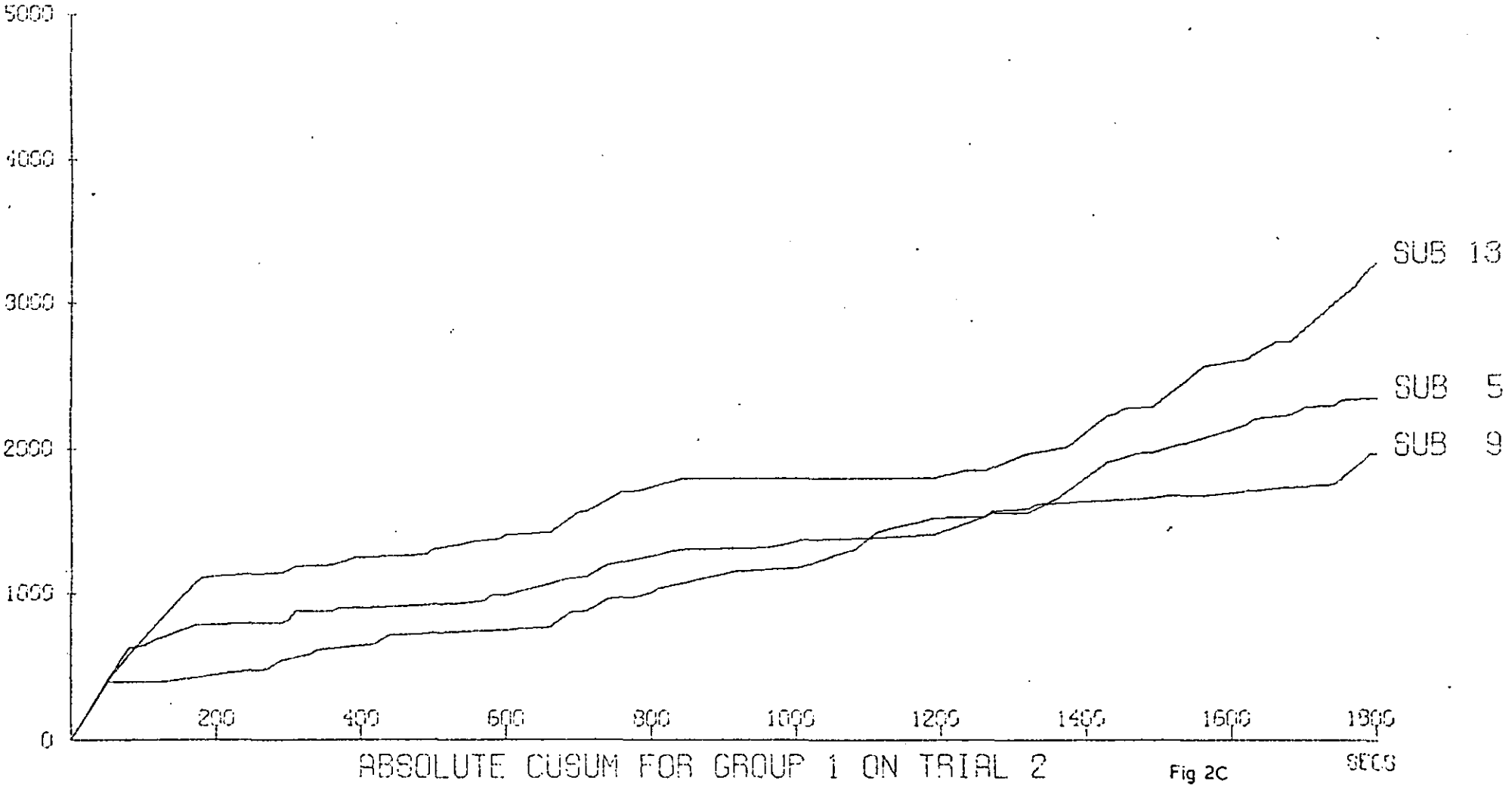
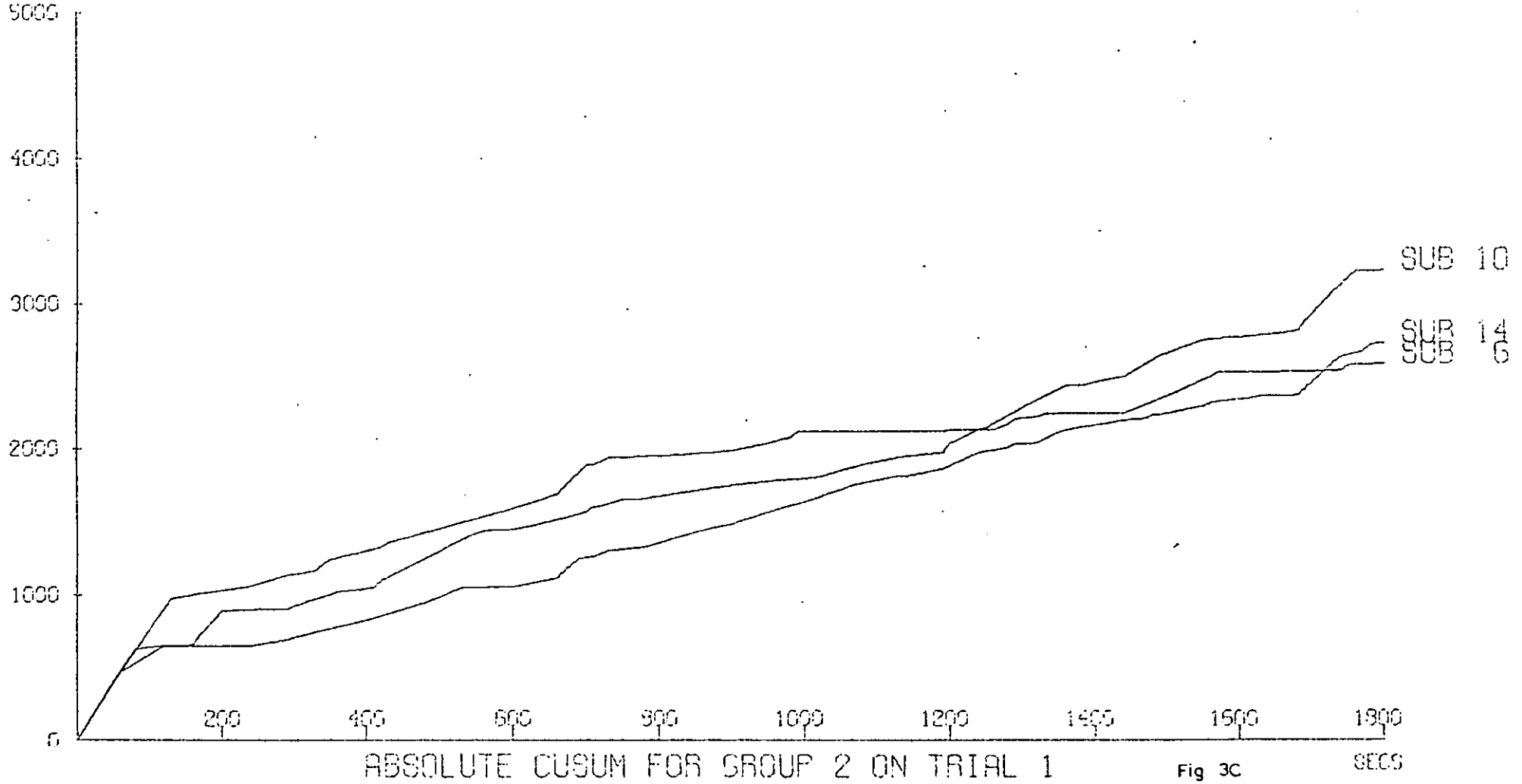

5000
4000
3000
2000
1000
0
200
400
600
800
1000
1200
1400
1600
1800
SECS
SUB 13
SUB 5
SUB 9
ABSOLUTE CUSUM FOR GROUP 1 ON TRIAL 2

Fig 2C



ABSOLUTE CUSUM FOR GROUP 2 ON TRIAL 1

Fig 3C

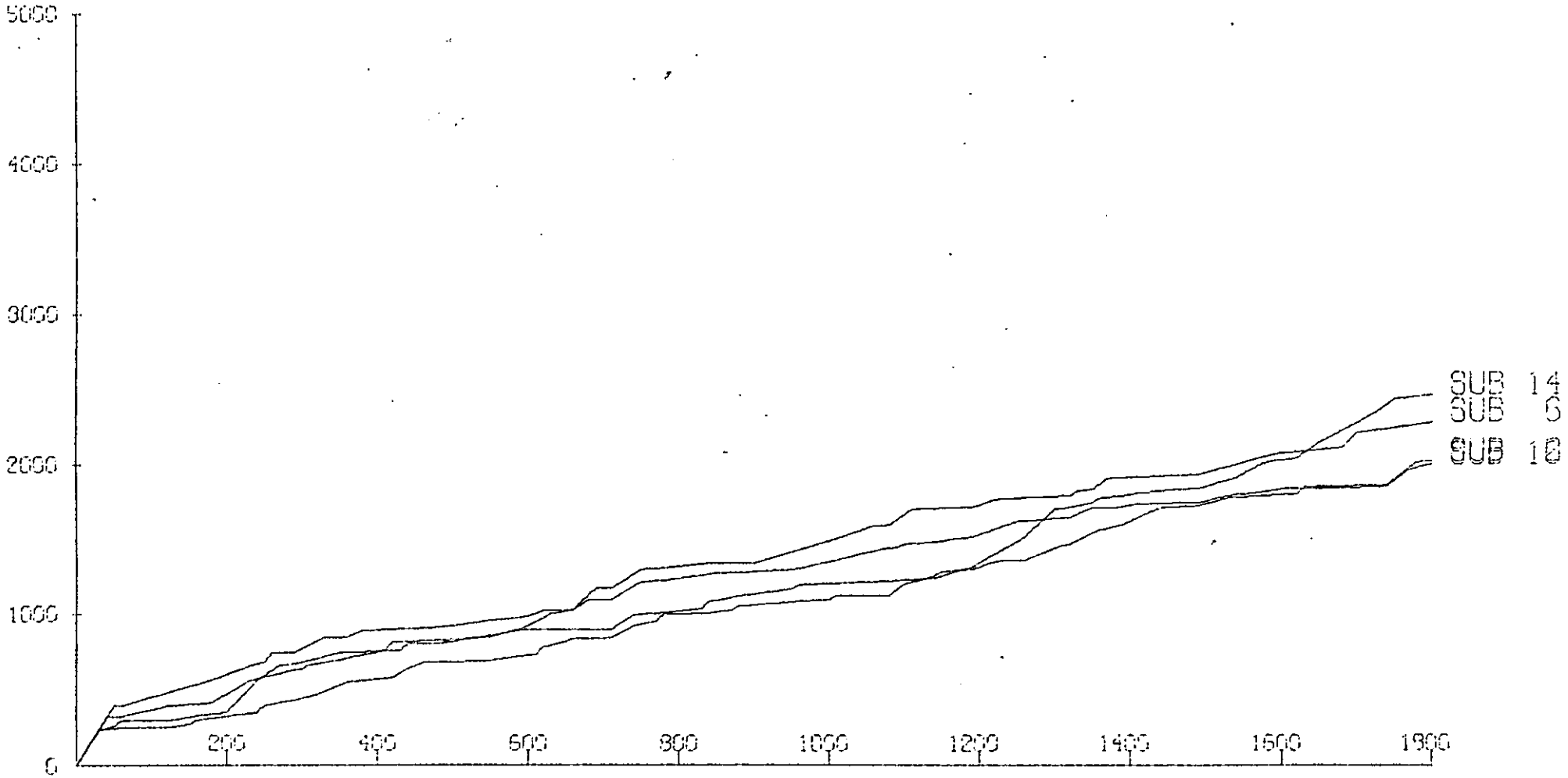

5000
4000
3000
2000
1000
0
200
400
600
800
1000
1200
1400
1600
1800
SECS
SUB 14
SUB 5
SUB 16
ABSOLUTE CUSUM FOR GROUP 2 ON TRIAL 2

Fig 4C

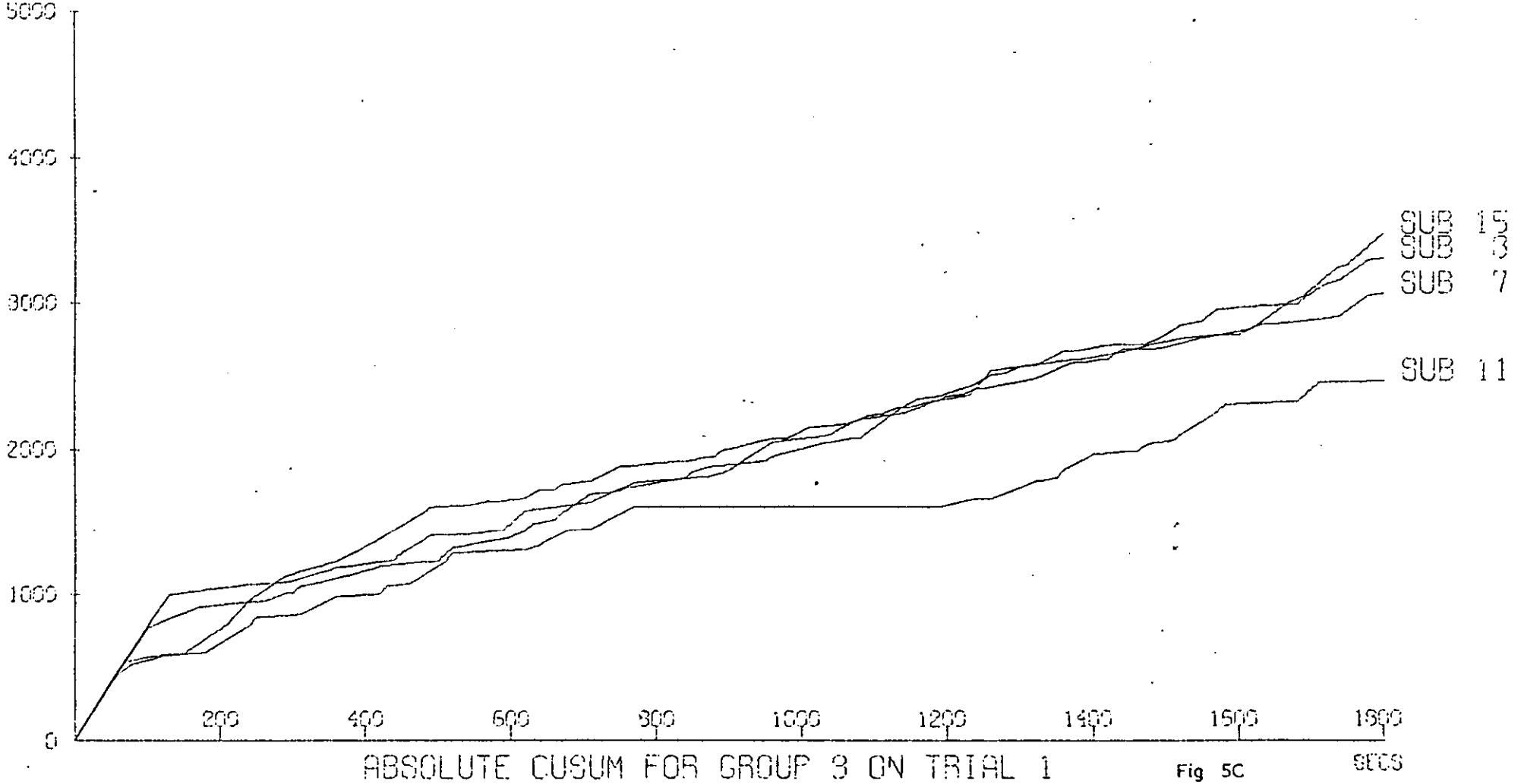
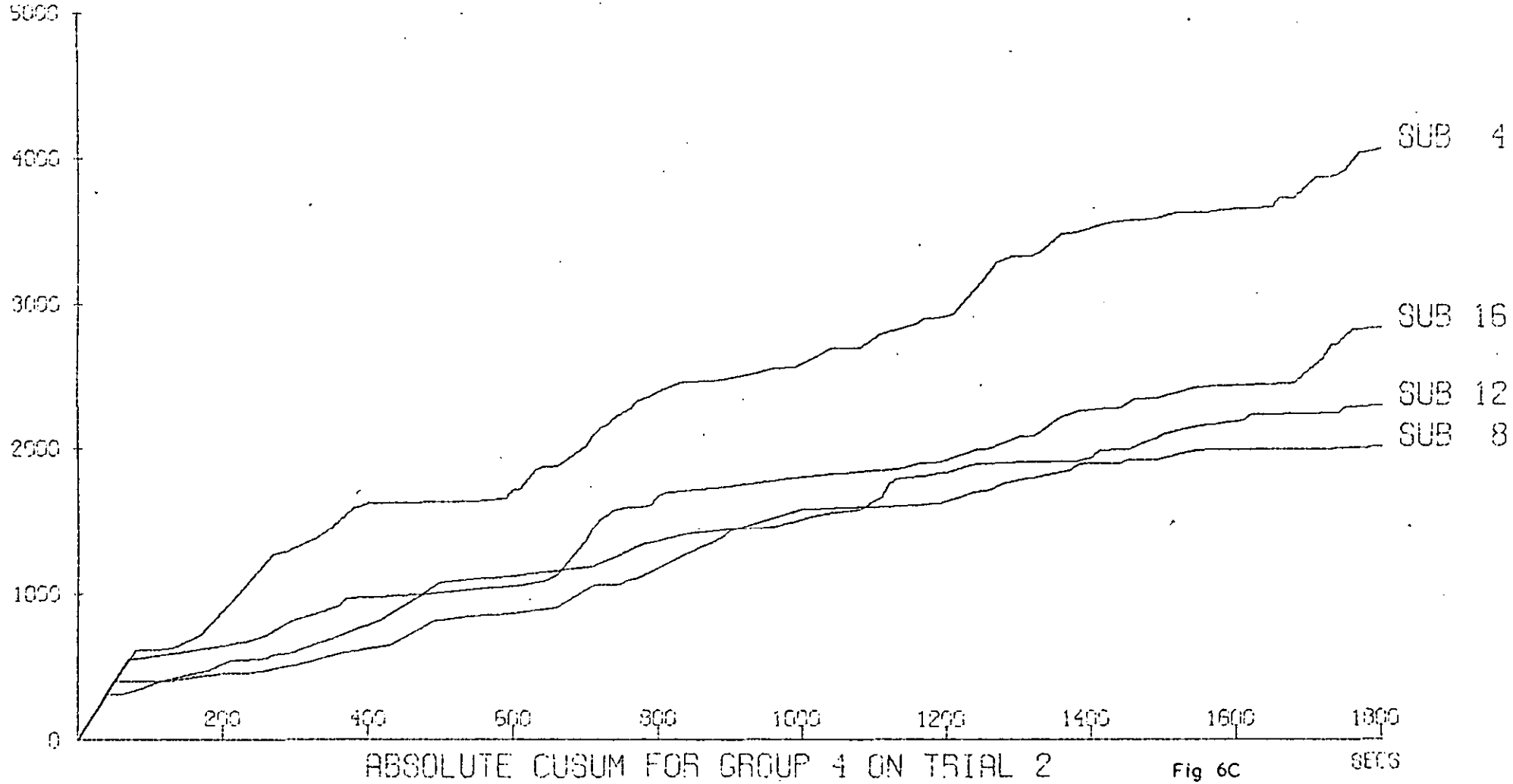

5000
4000
3000
2000
1000
0
200
400
600
800
1000
1200
1400
1600
1800
SECS
SUB 15
SUB 3
SUB 7
SUB 11
ABSOLUTE CUSUM FOR GROUP 3 ON TRIAL 1

Fig 5C

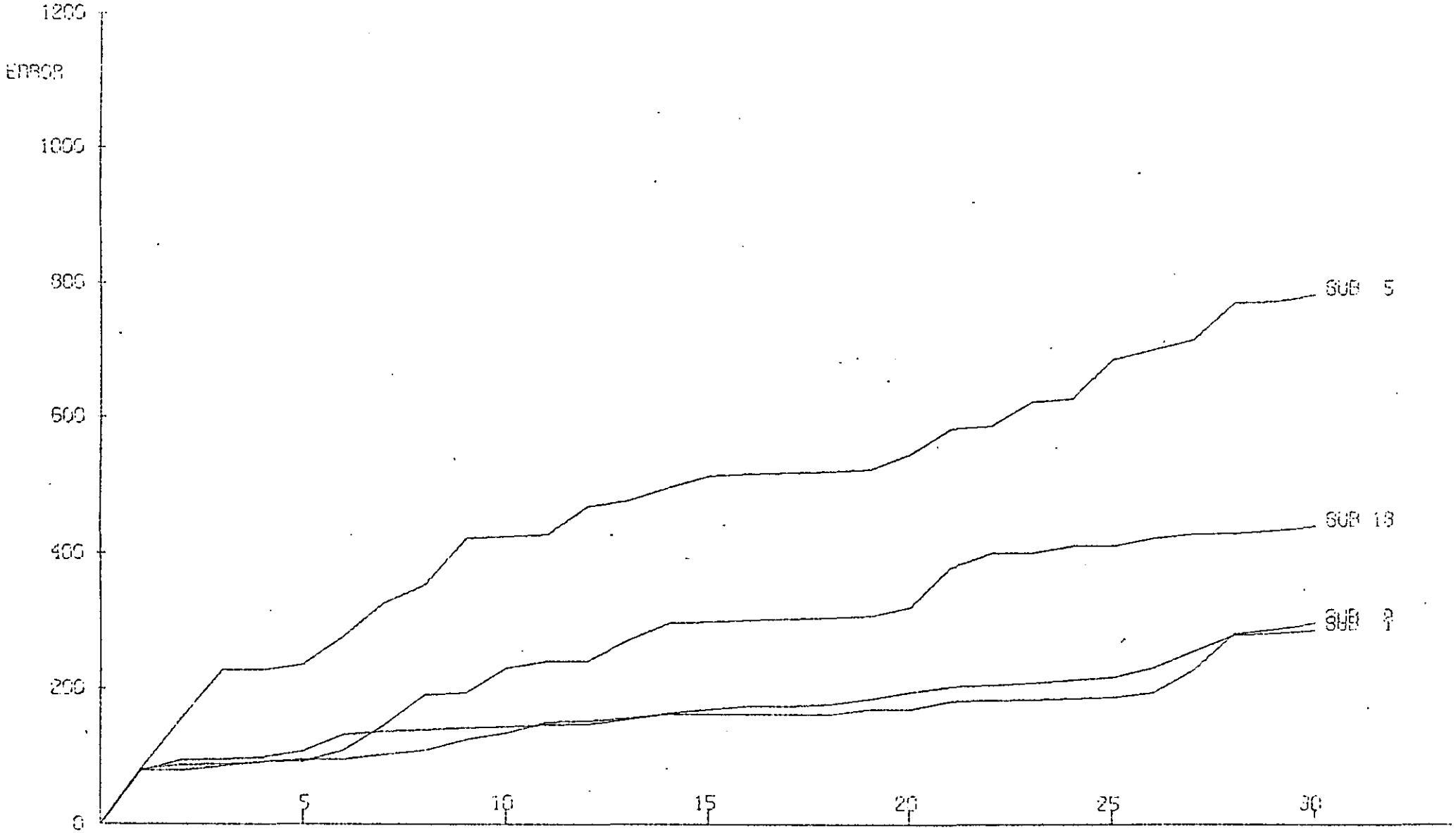


ABSOLUTE CUSUM FOR GROUP 4 ON TRIAL 2

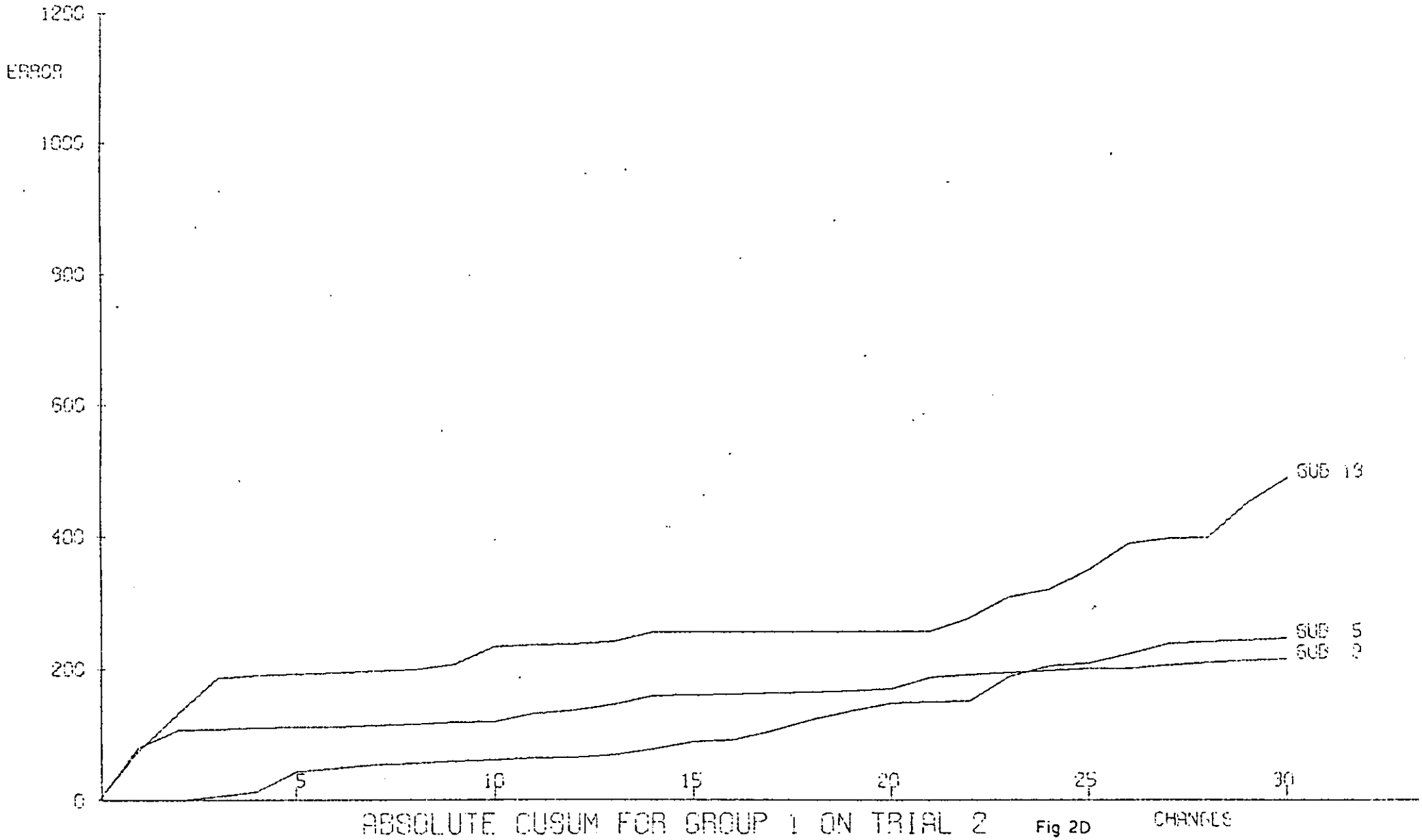
Fig 6C

ANNEX D

ABSOLUTE CUMULATIVE SUM ERROR SCORES AGAINST CHANGES



ABSOLUTE CUSUM FOR GROUP 1 ON TRIAL 1 Fig ID



ABSOLUTE CUSUM FOR GROUP 1 ON TRIAL 2 Fig 2D

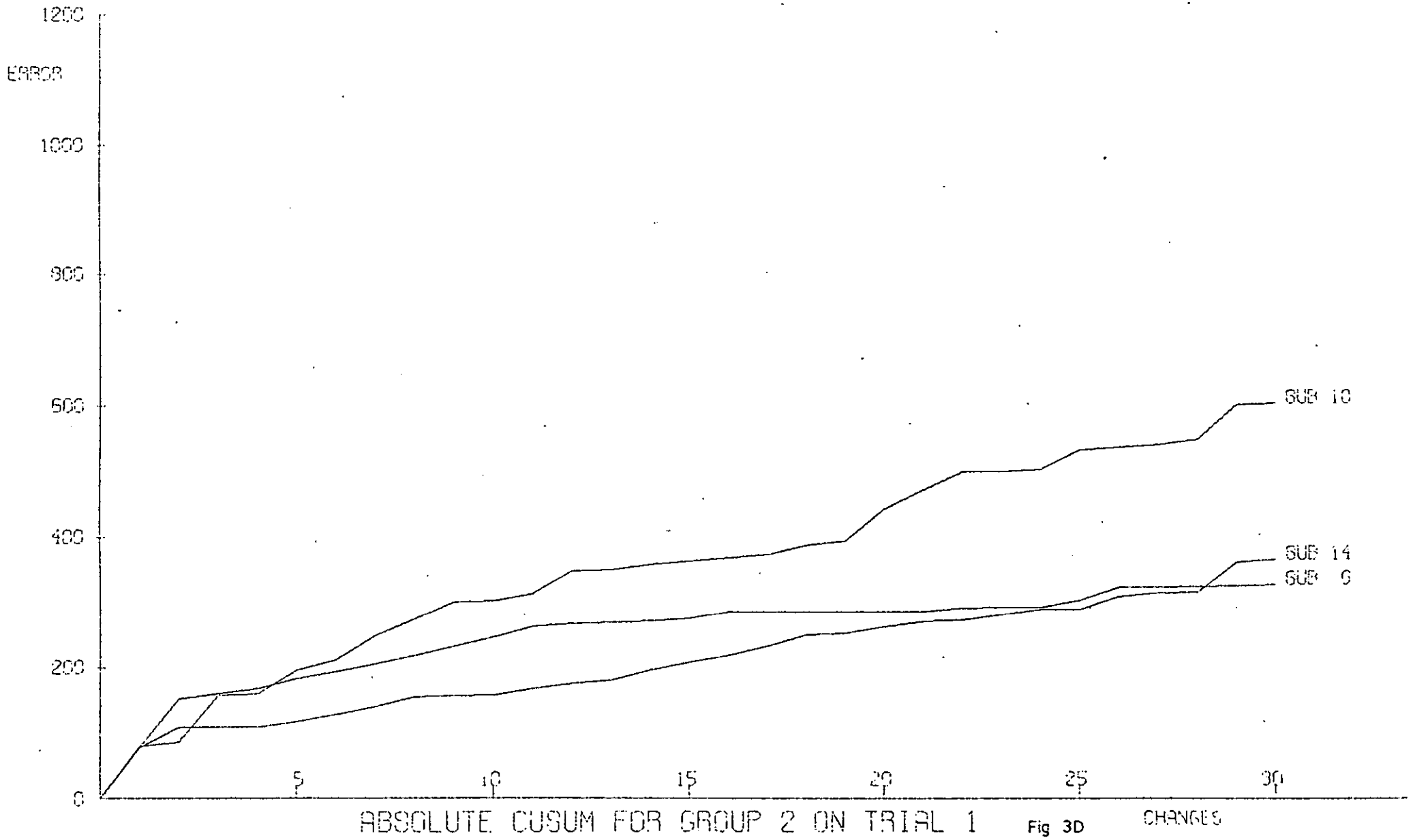
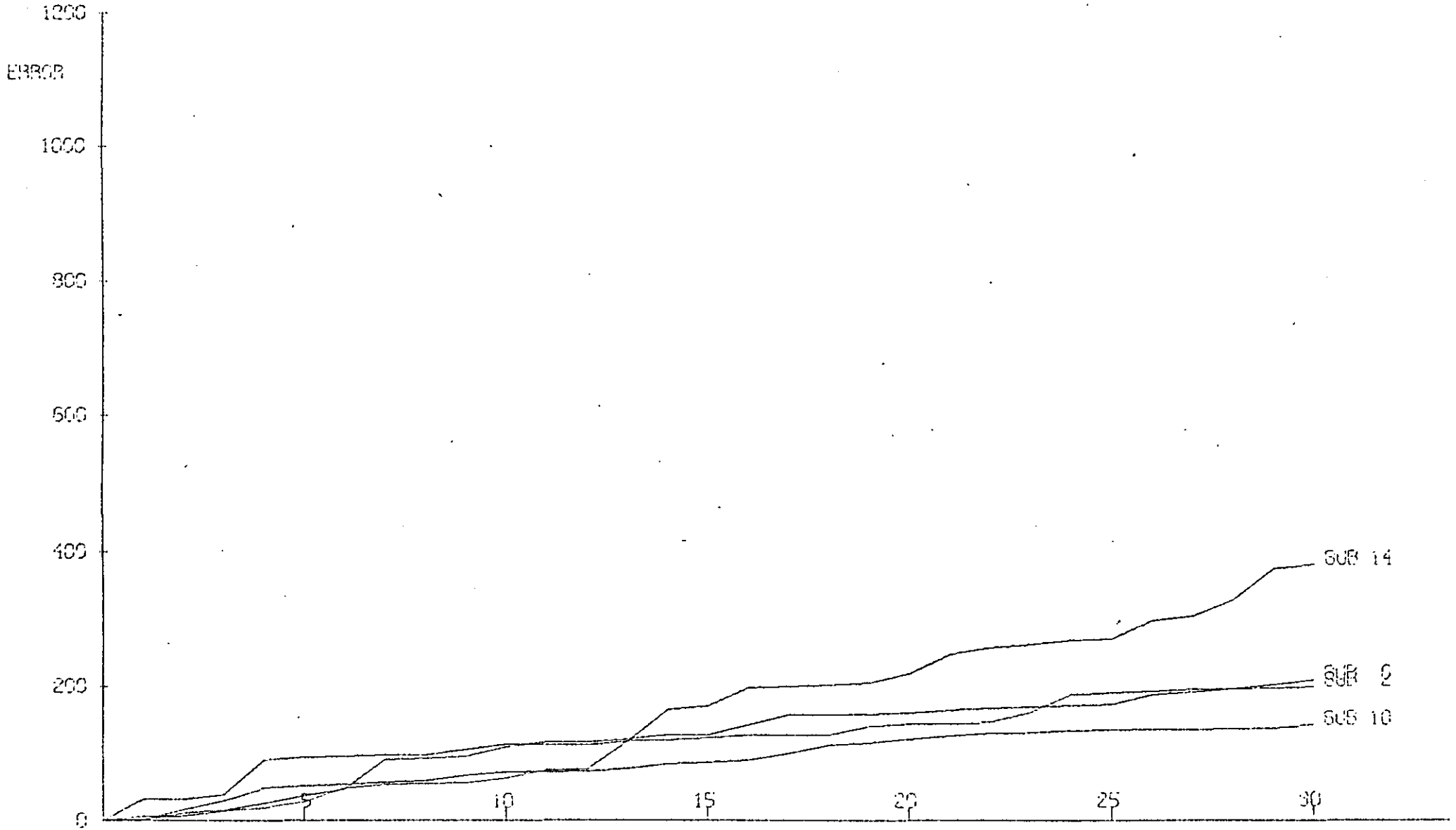
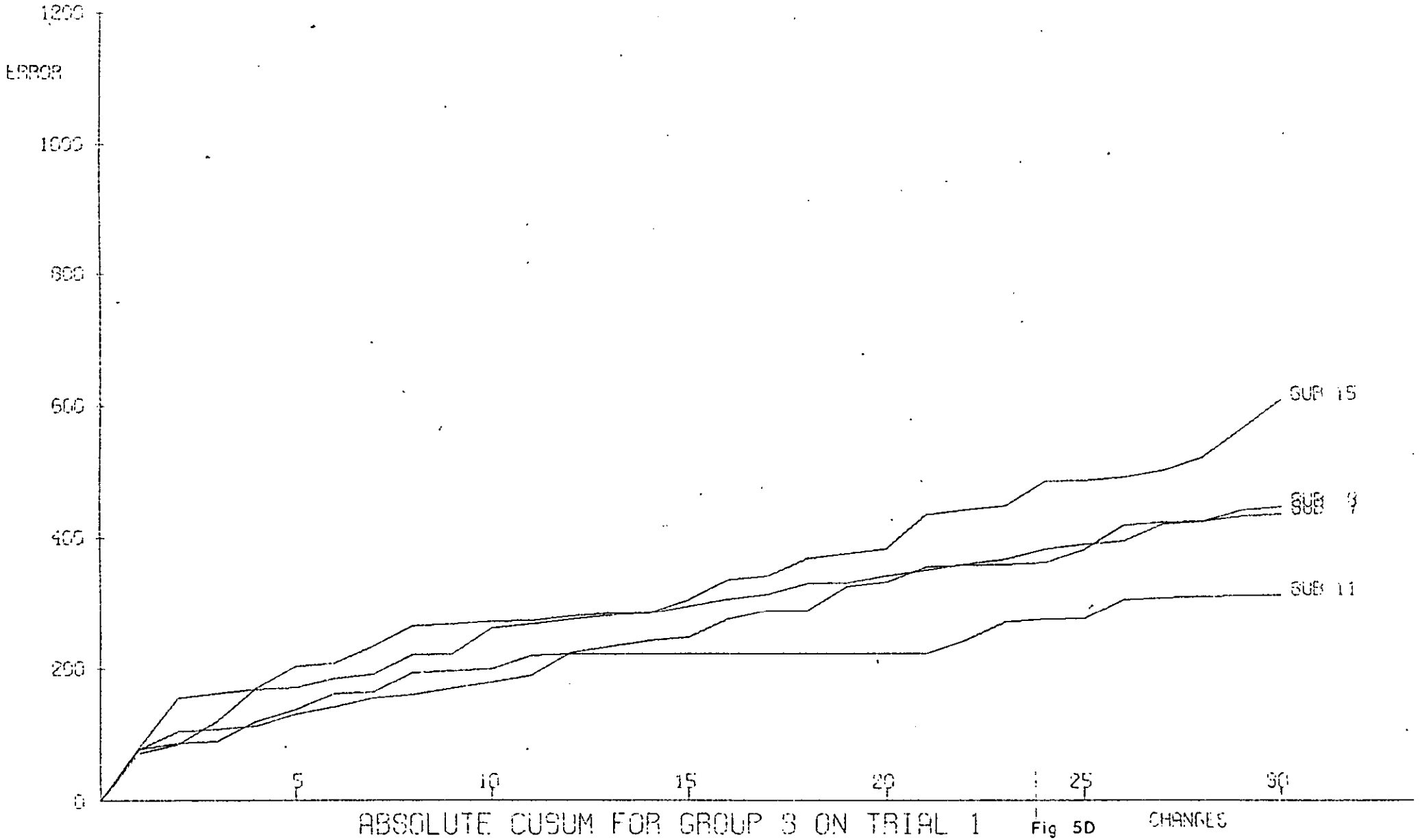

1200
ERROR
1000
800
600
400
200
0
5
10
15
20
25
30
SUB 10
SUB 14
SUB 9
ABSOLUTE CUSUM FOR GROUP 2 ON TRIAL 1
CHANGES

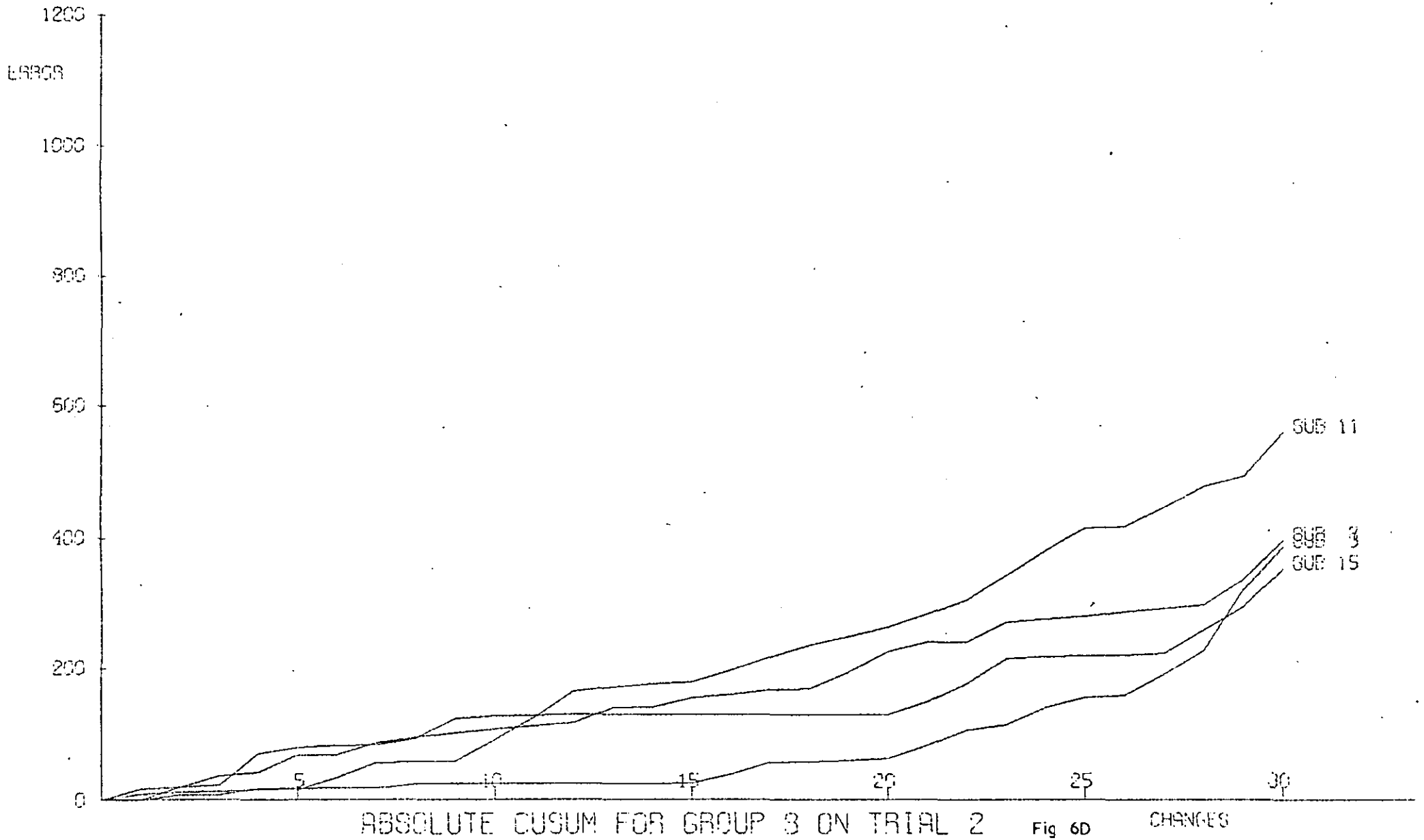
Fig 3D



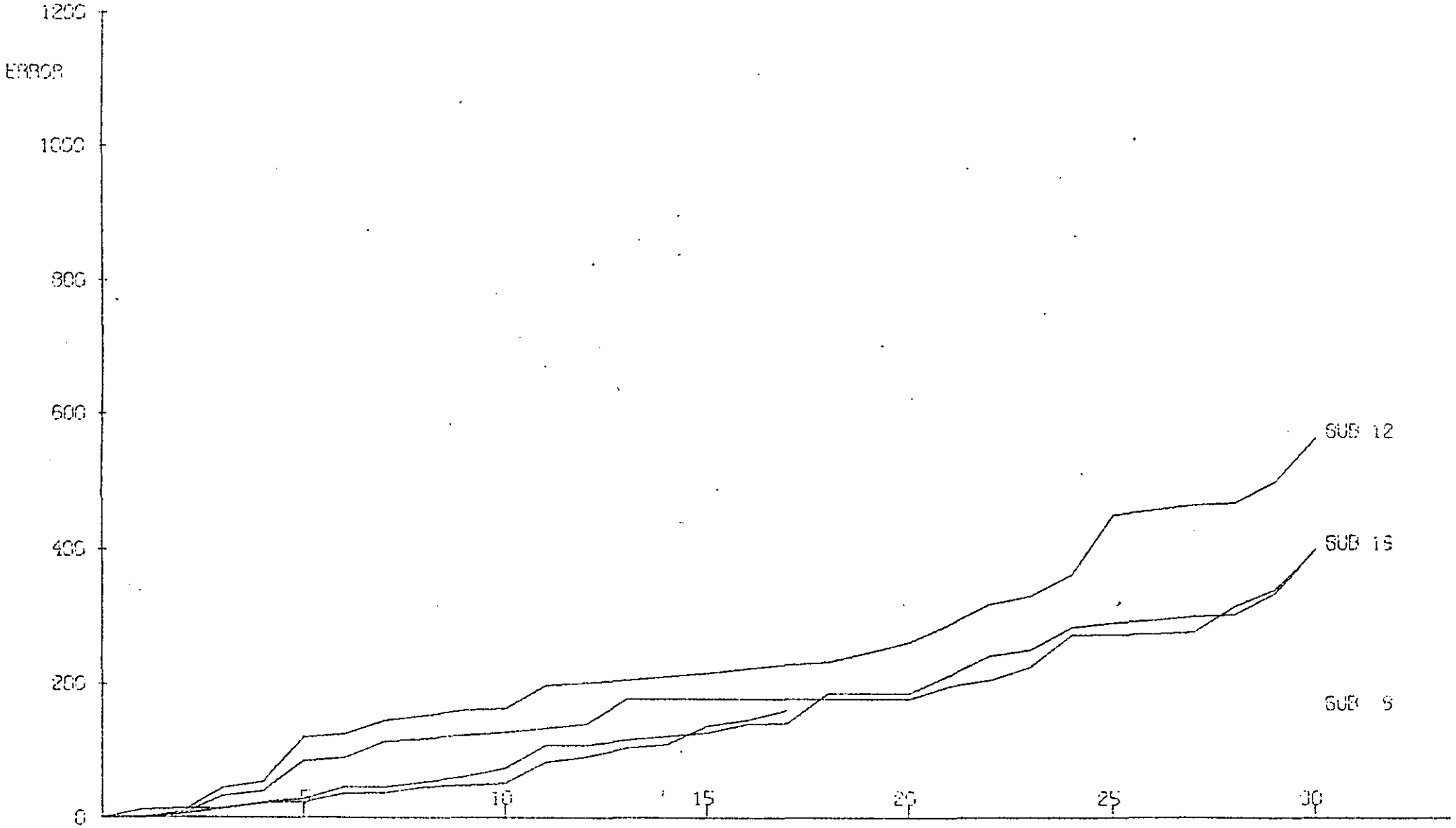
ABSOLUTE CUSUM FOR GROUP 2 ON TRIAL 2 Fig 4D



ABSOLUTE CUSUM FOR GROUP 3 ON TRIAL 1 Fig 5D



ABSOLUTE CUSUM FOR GROUP 3 ON TRIAL 2 Fig 6D



ABSOLUTE CUSUM FOR GROUP 4 ON TRIAL 1 Fig 7D

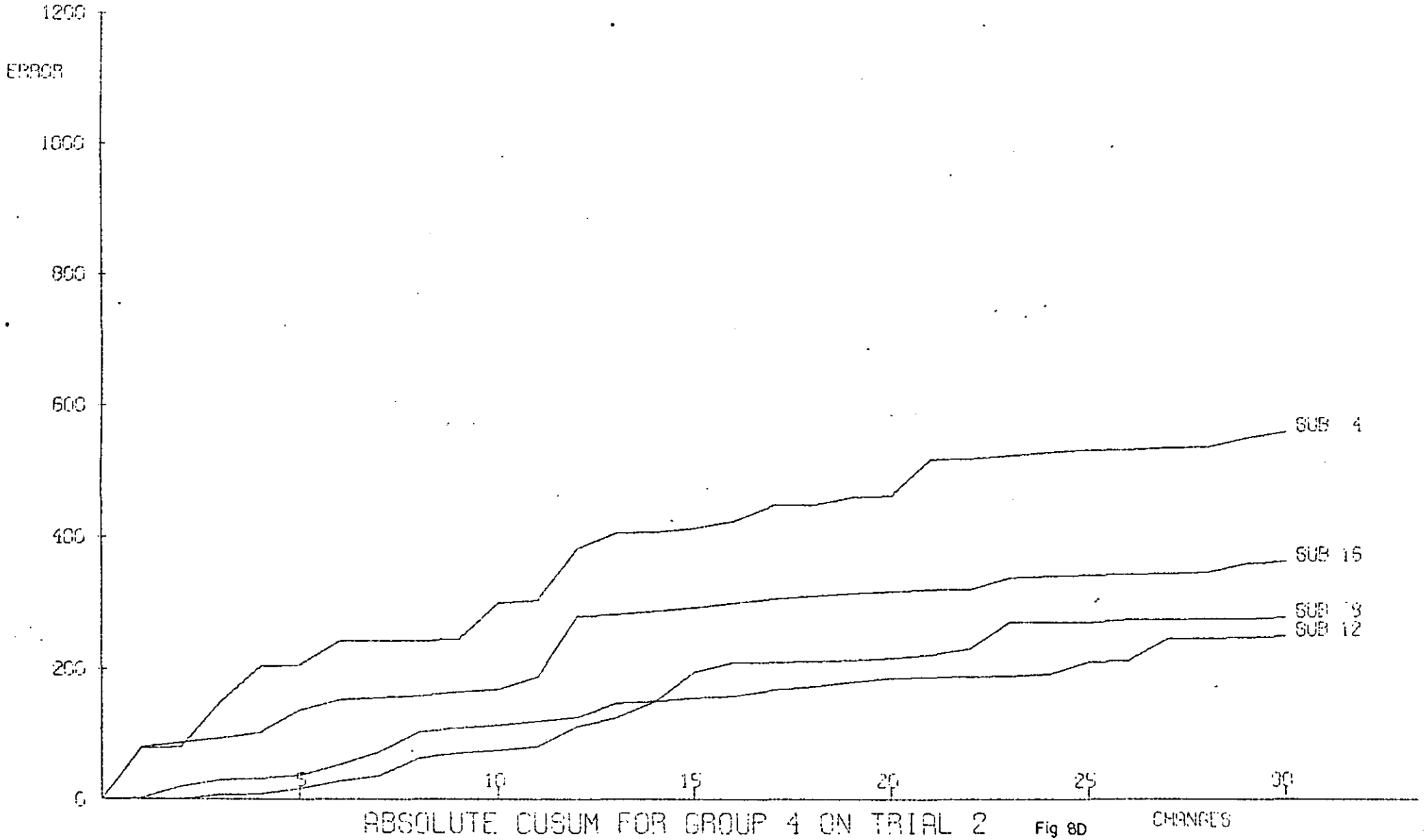
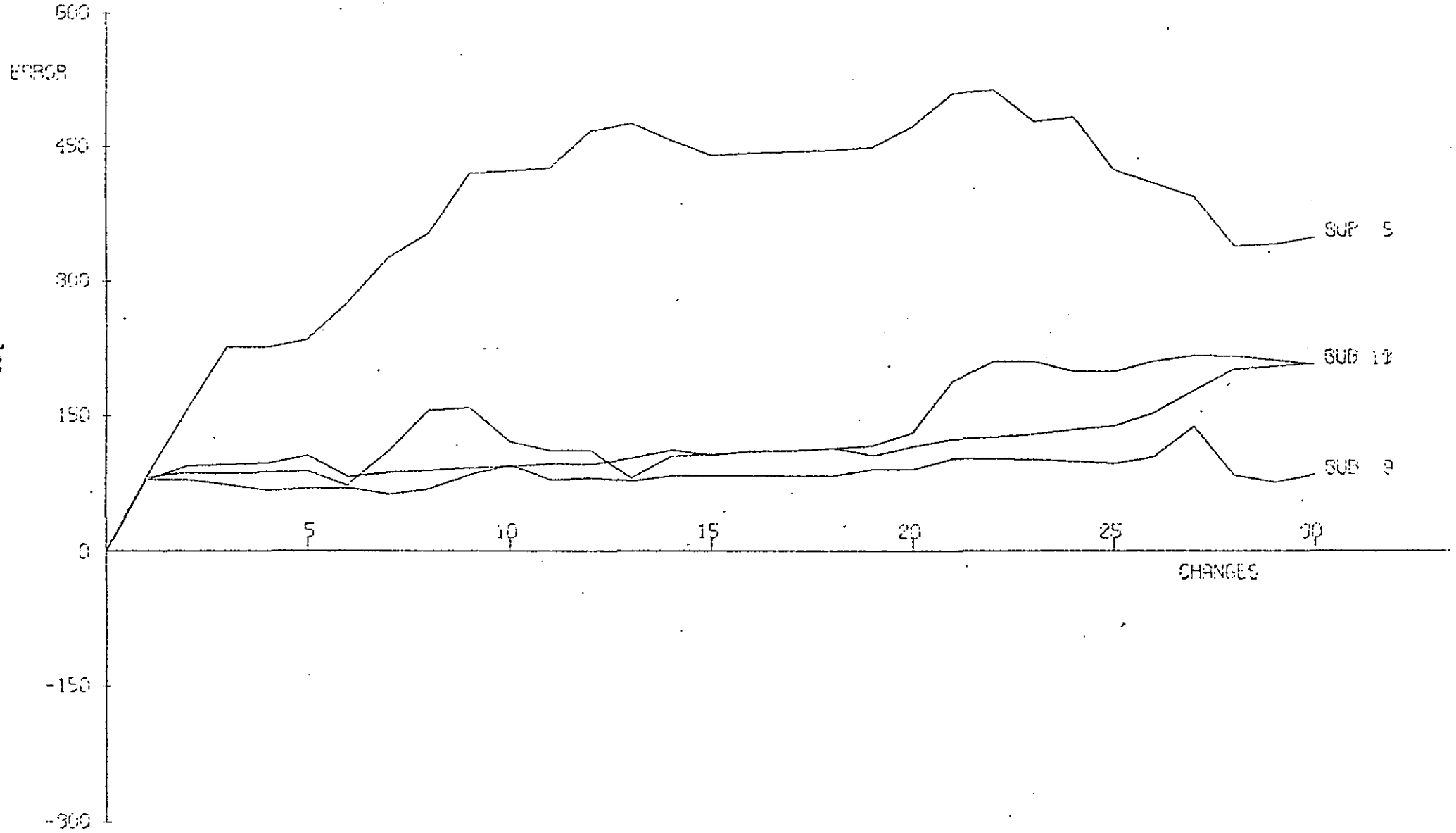

1200
ERROR
1000
800
600
400
200
0
5
10
15
20
25
30
SUB 4
SUB 15
SUB 13
SUB 12
ABSOLUTE CUSUM FOR GROUP 4 ON TRIAL 2
CHANGES

Fig 8D

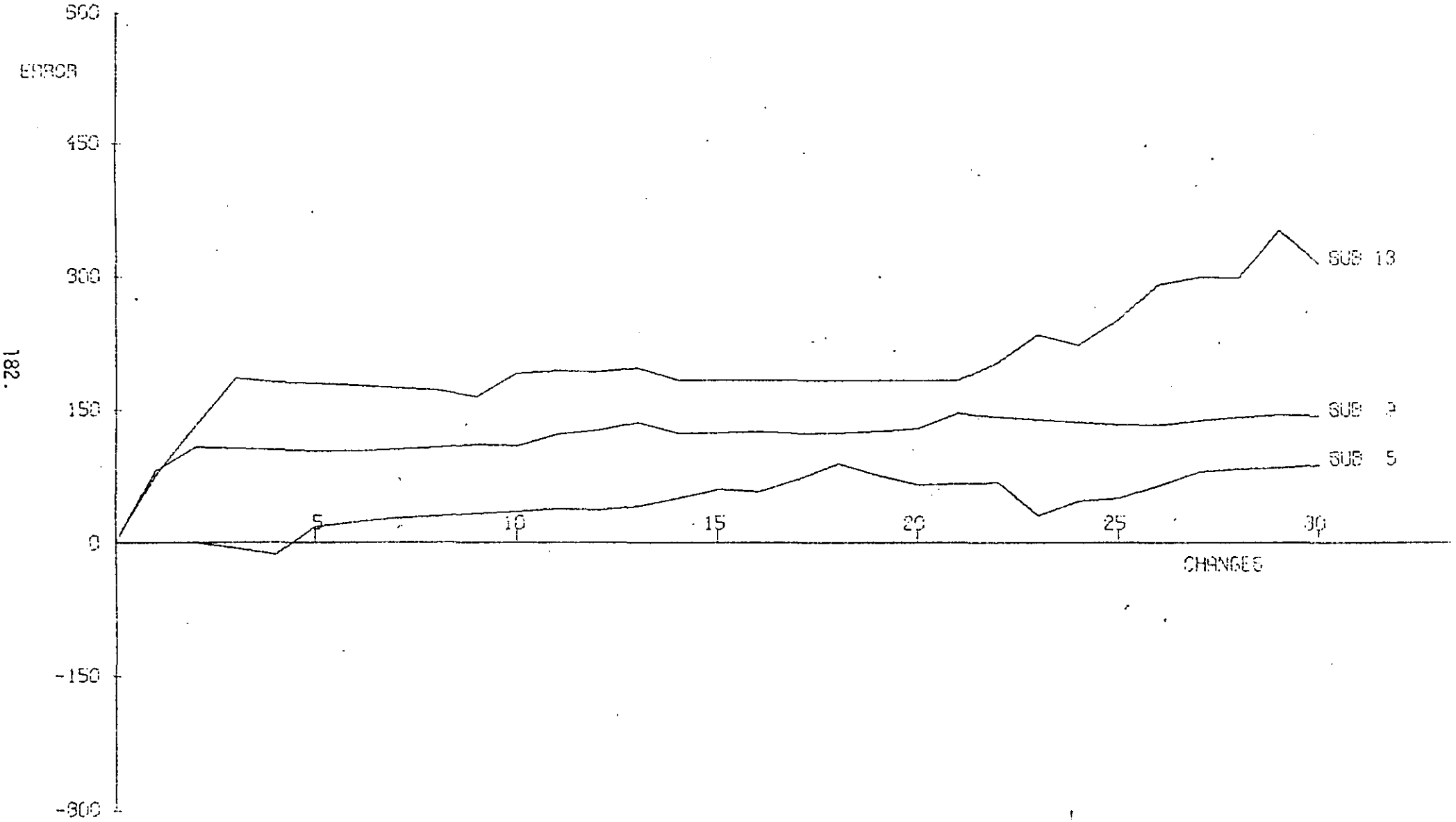
ANNEX E

RELATIVE CUMULATIVE SUM ERROR SCORES AGAINST CHANGES



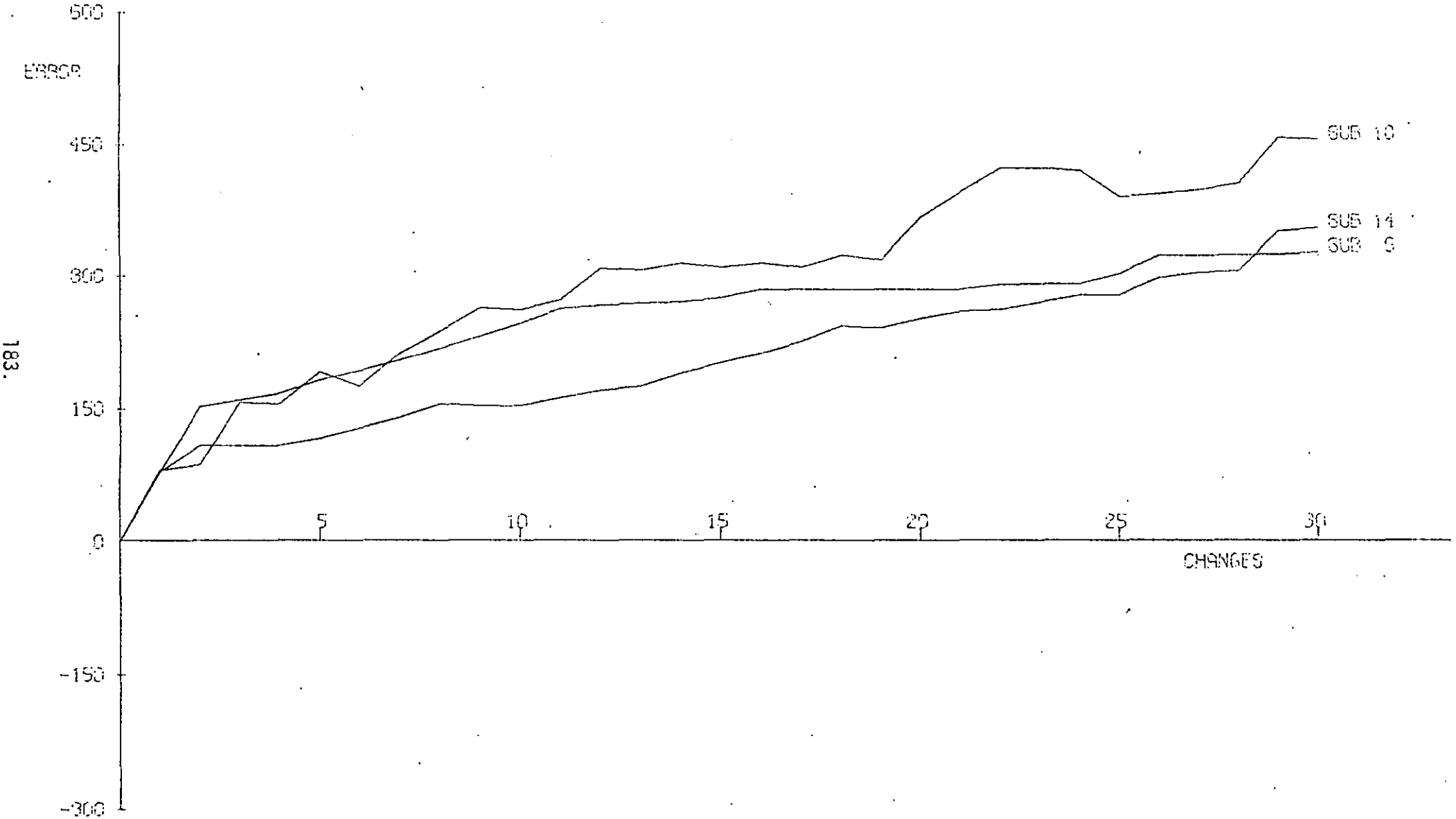
RELATIVE CUSUM FOR GROUP 1 ON TRIAL 1

Fig IE



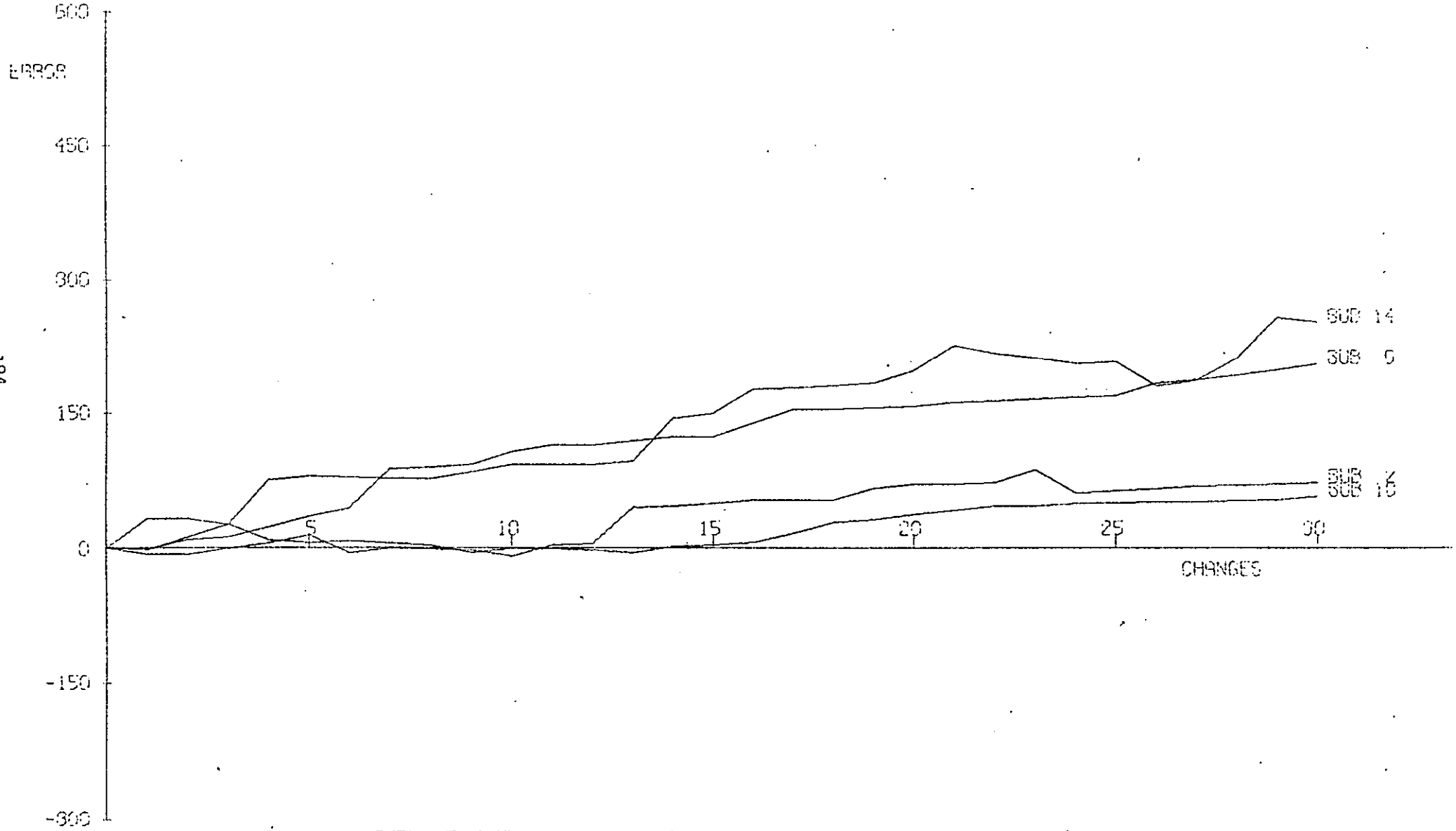
RELATIVE CUSUM FOR GROUP 1 ON TRIAL 2

Fig 2E



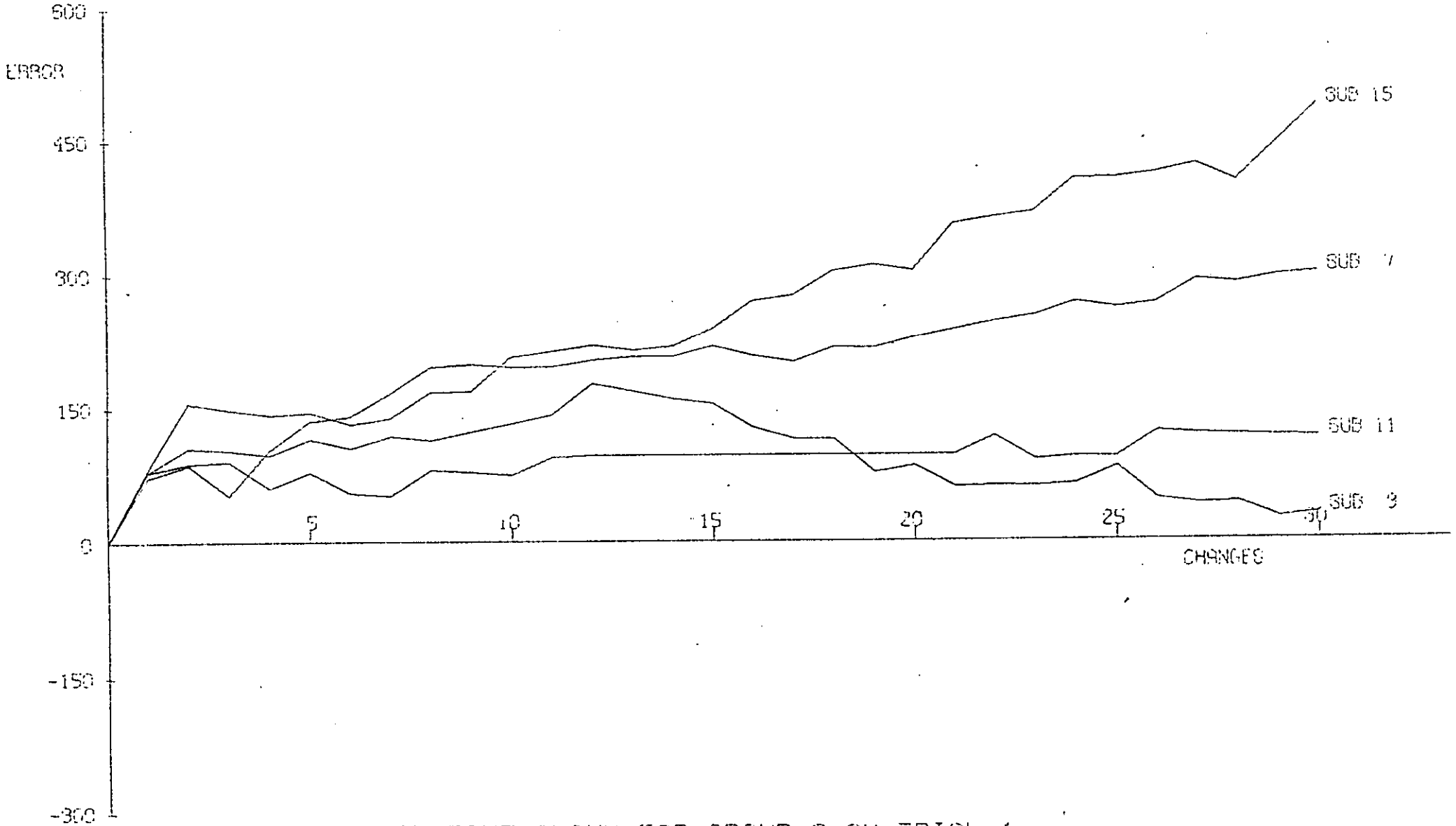
RELATIVE CUSUM FOR GROUP 2 ON TRIAL 1

Fig 3E



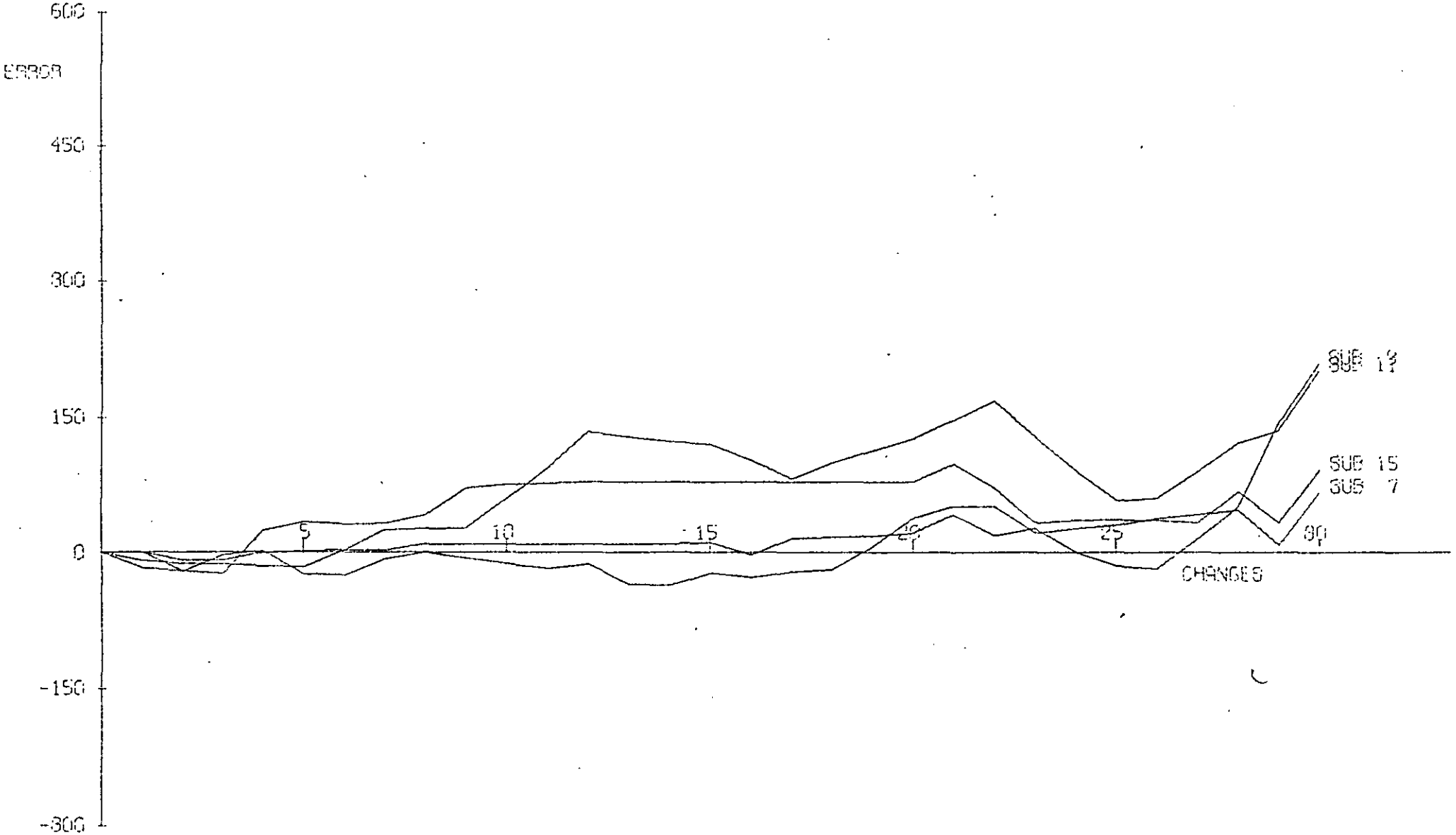
RELATIVE CUSUM FOR GROUP 2 ON TRIAL 2

Fig 4E



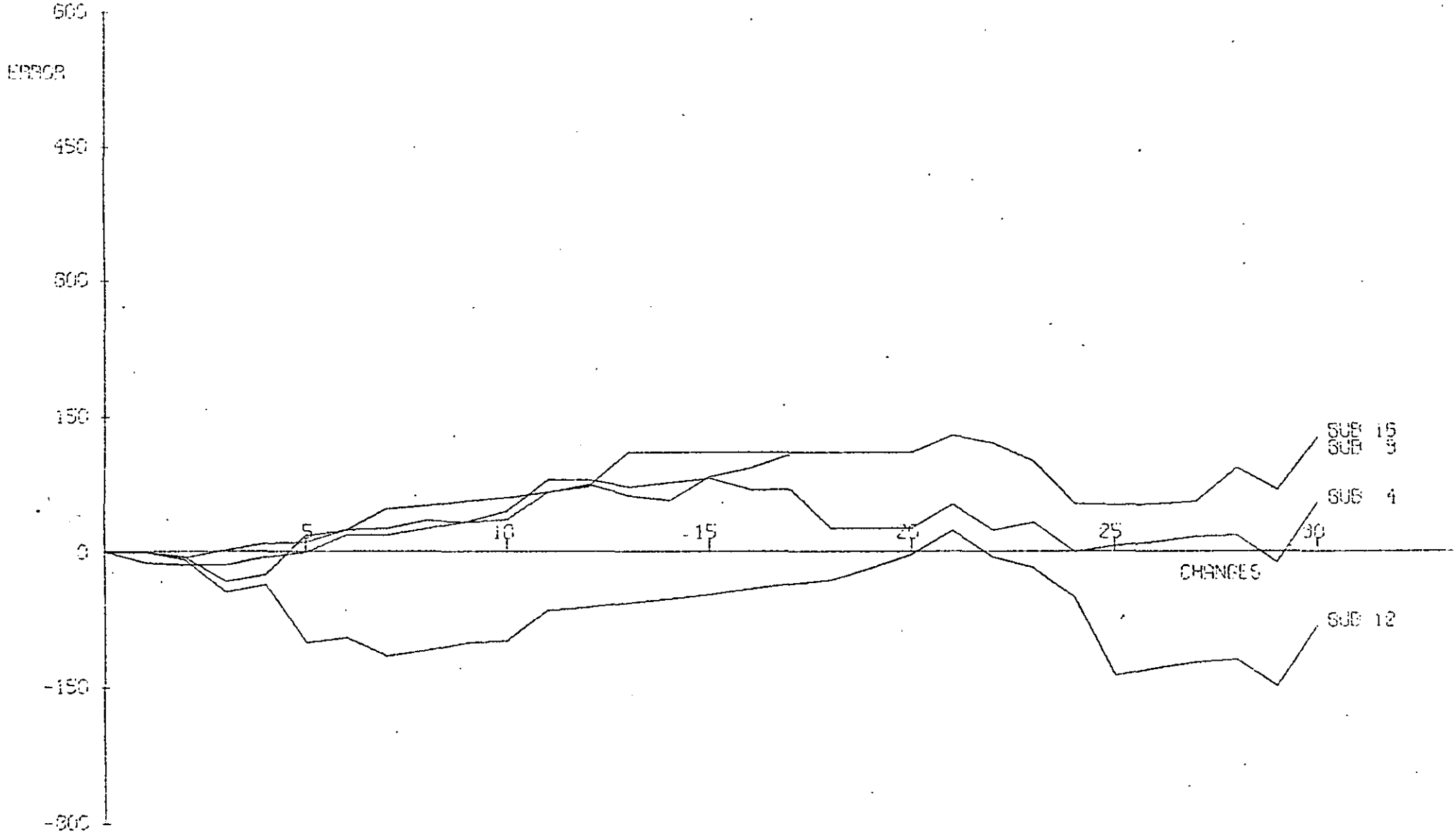
RELATIVE CUSUM FOR GROUP 3 ON TRIAL 1

Fig 5E



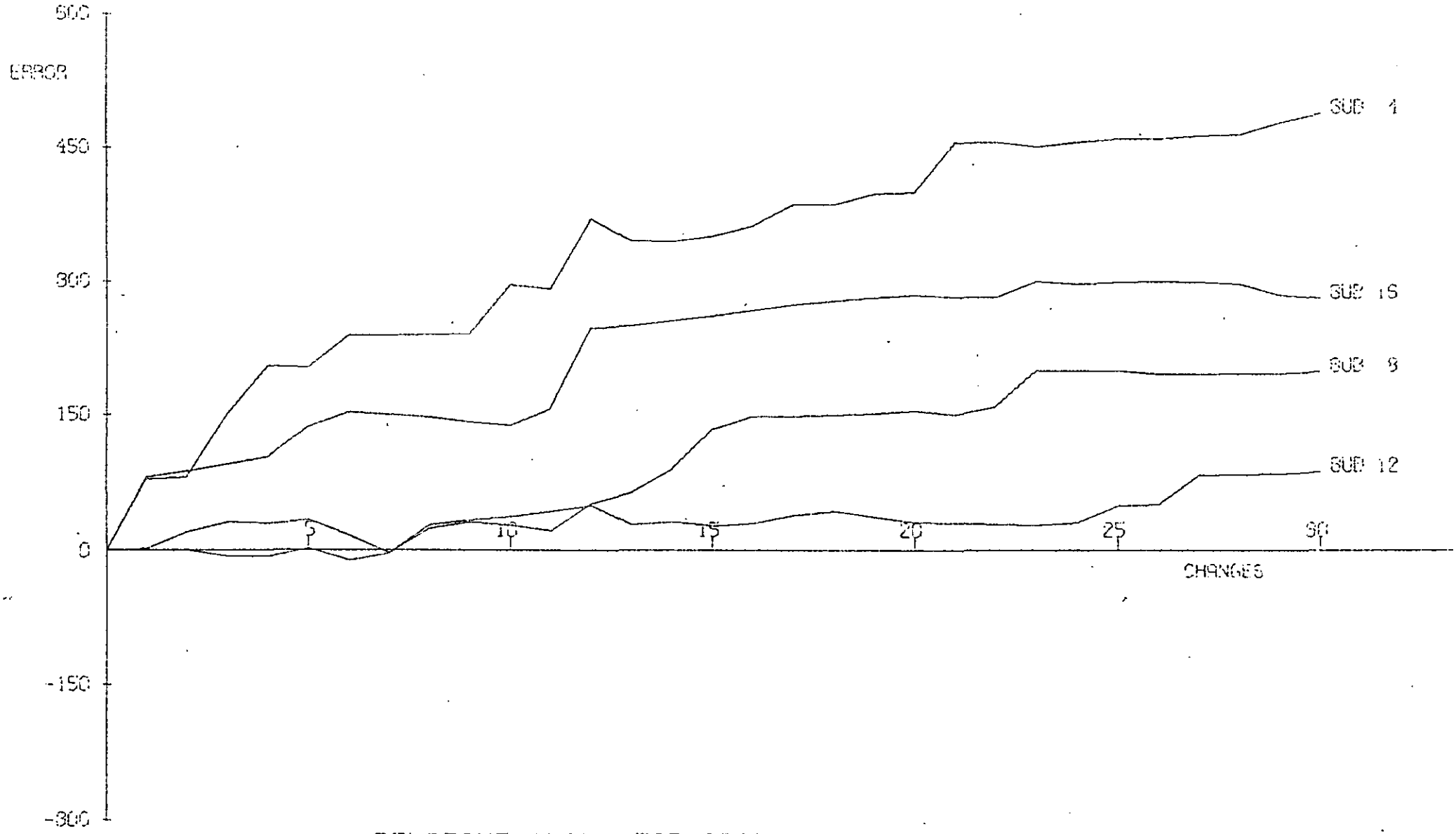
RELATIVE CUSUM FOR GROUP 3 ON TRIAL 2

Fig 6E



RELATIVE CUSUM FOR GROUP 4 ON TRIAL 1

Fig 7E



RELATIVE CUSUM FOR GROUP 4 ON TRIAL 2

Fig 8E