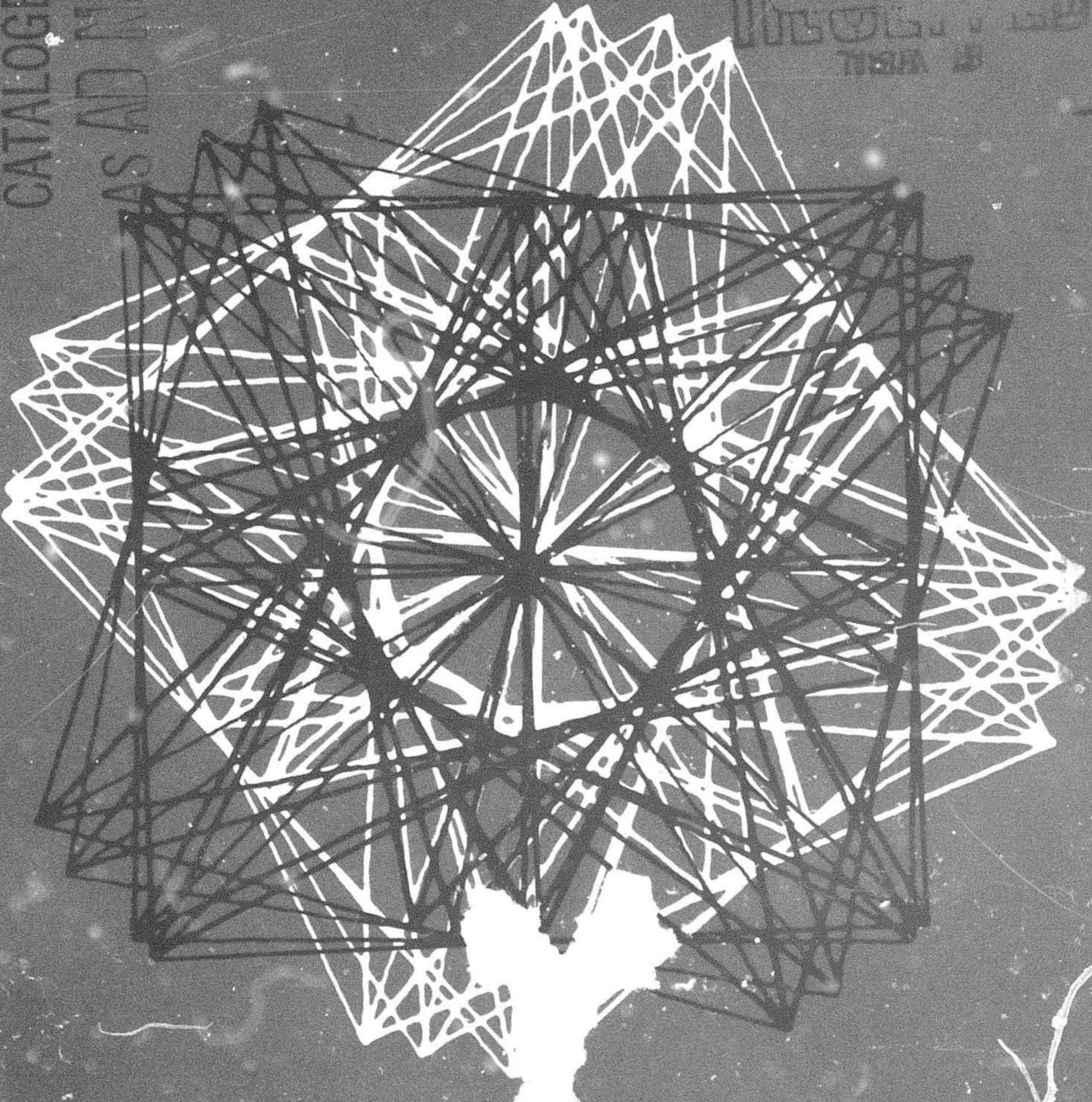
Massachusetts Institute of Technology Project MAC Progress to July 1964



Massachusetts Institute of Technology
Project MAC
545 Technology Square
Cambridge, Massachusetts
02139

Work reported herein was supported by Project MAC, an M.I.T. research program sponsored by the Advanced Research Projects Agency, Department of Defense, under Office of Naval Research Contract Number Nonr-4102(01). The primary support for some of this work came from the M.I.T. departments and laboratories participating in Project MAC, whose research programs are, in turn, sponsored by various government and private agencies.

This report was prepared on-line in the MAC computer system with the aid of the TYPSET and RUNOFF programs.

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

PERSONNEL
TO JULY 1, 1964

Administration	Committee On Information Processing
Prof. R. M. Fano - Director	Prof. P. M. Morse, Chairman
O. G. Selfridge - Associate Director	Dean G. S. Brown
R. G. Mills - Assistant Director for Administration	Prof. P. Elias
	Prof. R. M. Fano
	Prof. A. G. Hill
	Dean H. W. Johnson
	Prof. C. F. J. Overhage
	Prof. C. H. Townes, Provost
	Dean J. B. Wiesner

Faculty

D. M. B. Baumann	R. A. Howard	I. D. Pool
A. Bers	R. Y. Kain	A. H. Rosenfeld
J. M. Biggs	R. Kaplow	A. L. Samuel
D. G. Bobrow	D. J. Kuck	R. L. Schiffman
R. W. Brockett	J. D. Little	T. B. Sheridan
D. C. Carroll	C. L. Liu	S. G. Simpson
S. A. Coons	R. D. Logcher	L. D. Smullin
F. J. Corbato	L. A. Lombardi	P. O. Soelberg
J. B. Dennis	M. L. Manheim	T. G. Stockham
Z. M. Elias	S. P. Mauch	H. M. Stommel
J. C. Emery	R. T. McLaughlin	G. M. Sturman
R. M. Fano	C. L. Miller	H. M. Teager
M. Greenberger	M. L. Minsky	J. R. Walton
J. M. Heinz	P. M. Morse	J. Weizenbaum
	H. M. Paynter	

Research Associates and Instructors

E. L. Glaser	G. R. Murray	J. H. Saltzer
A. M. Hershdorfer	D. Roos	V. H. Yngve
C. C. Luckham		

Research Staff

J. A. Arnow	E. G. Guttmann	H. C. Peterson
R. U. Bayles	D. R. Haring	R. B. Polansky
F. Belvin	T. N. Hastings	L. H. Pouzin
J. Bex	M. C. Henneman	S. L. Rosenbaum
R. J. Bigelow	J. M. Hodnick	D. T. Roos
C. M. Bosche	R. E. Holz	A. Rutchka
E. J. Campbell	B. A. Johanson	A. J. Saltalamacchia
R. J. Creasy	M. S. Kannel	G. G. Schroeder
M. M. Daggett	M. M. Kessler	O. G. Selfridge
R. C. Daley	E. M. Kliman	J. K. Smith
S. D. Dunten	P. T. Ladd	M. V. Solomita
D. J. Edwards	C. A. Lang	R. H. Stotz
F. J. Eppling	R. Leopold	P. A. Sullivan
R. S. Fabry	J. L. McDonald	H. F. Tonsing
G. M. Fratar	C. Marceau	J. F. Walsh
C. Garman	R. G. Mills	J. E. Ward
R. M. Graham	J. F. Nolan	O. C. Wright
U. F. Gronemann	M. M. Pennell	D. B. Yntema

Resident Guests

D. P. Bovet	B. T. Fox	I. C. Pyle
C. W. Bower	J. Kennedy	H. W. Spencer
R. K. Coe	J. J. Martyniak	

12

Research Assistants

L. N. Beckreck	G. C. Ling	J. R. Roy
D. Beltran	D. T. Llewellyn-Jones	F.J. Russo
J. R. Brach	D. C. Luckham	A. L. Scherr
J. E. Burke	W. A. Martin	R. L. Schwenke
S. T. Coleman	R. M. Merrill	T. W. Schwenke
G. C. Everest	A. J. Miller	H. L. Selesnick
S. C. Finkelstein	J. R. Miller III	L. L. Selwyn
H. C. Frazier	J. D. Mills	S. L. Strong
A. Gabriellan	J. Moses	J. H. Suhrbier
R. V. Goodman	H. M. Nanavati	L. C. Teague
K. L. Groninger	L. M. Norton	C. C. Tillman
D. R. Hamann	R. E. Oaklund	E. C. VanHorn
T. P. Hart	R. P. Parmelee	R. A. Walter
M. G. Hurwitz	W. M. Pecknold	J. W. Weber
E. L. Ivie	N. R. Patel	M. R. Weinberger
M. M. Jones	J. H. Ransom	A. D. Weiss
A. H. Kessler	B. Raphael	P. A. Weiselman
J. R. Kramer	J. E. Rodriguez	R. L. Welsh
W. H. Linder	T. G. Roseman	D. U. Wilde
J. L. Linderman	R. C. Rosenberg	B. L. Wolman

Graduate Students

B. Bloom	L. S. Kanodia	L. L. Roos
J. E. Burke	J. Keller	H. M. Schneider
J. E. Dailey	C. Kim	H. S. Schwenk
S. R. Ditmeyer	F. F. Lee	J. R. Sklar
E. A. Feustel	M. Levin	R. W. Spitz
R. C. Gammill	N. W. Luttrell	J. P. Svenne
D. G. Gladding	J. Morris	W. Teitelman
G. A. Gorry	D. B. Murton	M. Wantman
E. C. Haines	J. D. Nagle	S. H. Whitelaw
M. L. Hamilton	F. E. Perkins	P. A. Wieselman
T. Hoffman	S. L. Popkin	R. A. Wiggins
W. F. Johnson	K. E. Reinschmidt	

Undergraduates and Part-Time Assistants

W. C. Albertson	H. D. Levin	K. Rosenfeld
M. F. Brescia	P. C. Lynch	D. Sacks
G. Y. Chang	B. K. Marvin	W. D. Selles
T. W. Eggers	S. G. Mazzotta	P. Sexton
H. T. French	D. D. Moran	R. South
G. C. Gregory	S. Nelson	R. G. Tepper
R. M. Hodges	J. D. Price	R. Thurber
J. T. Holloway	R. E. Raffo	T. H. VanVleck
J. Howard	G. L. Rosen	J. Wagner
J. Krakauer		

Technical Assistants

G. Blum	C. H. Willson	A. W. Wilson
R. A. Humphrey		

Technician

R. Porcaro

Operating Staff

G. Andrews	J. Eaton	G. Noseworthy
R. Borsini	D. Ferguson	M. Pagliarulo
R. Cerrato	R. B. Hart	E. Reardon
G. Dardis	T. Martin	J. Spera
T. Dattilio	R. McNamara	J. Walker
R. Jegan	D. Moore	

Administrative and Supporting Staff

T. L. Bergeron	C. King	D. Scanlon
E. Gannon	F. Lee	L. Silvestro
M. Grourke	J. Pinella	C. Solomon
J. Kinasewich	C. Sarandrea	

x

PREFACE

Project MAC was organized at the Massachusetts Institute of Technology in the spring of 1963 for the purpose of conducting a research and development program on Machine-Aided Cognition and Multiple-Access Computer systems. It operates under contract with the Office of Naval Research, acting on behalf of the Advanced Research Projects Agency of the Department of Defense.

The broad goal of Project MAC is the experimental investigation of new ways in which on-line use of computers can aid people in their individual intellectual work, whether research, engineering design, management, or education. One envisions an intimate collaboration between man and computer system in the form of a real-time dialogue where both parties contribute their best capabilities. Thus, an essential part of the research effort is the evolutionary development of a large, multiple-access computer system that is easily and independently accessible to a large number of people, and truly responsive to their individual needs. The MAC computer system is a first step in this direction and is the result of research initiated several years ago at the MIT Computation Center. Technical Report MAC-TR-12 describes the system as of June, 1964.

Project MAC was organized in the form of an interdepartmental, interlaboratory "project" to encourage widespread participation from the MIT community. Such widespread participation is essential to the broad, long-term project goals for three main reasons: exploring the usefulness of on-line use of computers in a variety of fields, providing a realistic community of users for evaluating the operation of the MAC computer system, and encouraging the development of new programming and other computer techniques in an effort to meet specific needs.

Faculty, research staff, and students from fourteen academic departments and four interdepartmental research laboratories are participating in Project MAC. For reporting purposes, they are divided into thirteen groups, whose names correspond in many cases to those of MIT schools, departments and research laboratories. Some of the groups deal with research topics that fall under the heading of computer sciences; others with research topics which, while contributing in a substantive way to the goals of Project MAC, are primarily motivated by objectives outside the computer field.

The purpose of this Progress Report is to outline the broad spectrum of research being carried out as part of Project MAC. Some of the research is cosponsored by other governmental and private agencies, and its results are described in journal articles, reports and theses emanating from the various MIT departments and laboratories participating in Project MAC. Such publications are listed in Appendix A of the report. Internal memoranda of Project MAC are listed in Appendix B, and Project MAC Technical Reports are listed on the inside of the back cover.

Robert M. Fano
Cambridge, Massachusetts

ADMINISTRATION AND SERVICES

Measurements on the Time-Sharing System Performance

<u>Measurements on the Time-Sharing System Performance</u> - Richard G. Mills

Two kinds of statistics on the time-sharing system are being systematically gathered: the first is oriented to the system and should yield information about the efficient use of the central computer, and the second is calculated to measure the degree of satisfaction of user demands. In the system-oriented statistics, running time and program size are modes for the classification of computer use. Within each bin determined by this two dimensional classification are recorded the number of entries and the computed averages of running time and of elapsed time and other statistics.

User-oriented statistics are produced in a different way. The statistics gathering program samples the state of the supervisor at frequent intervals, often of five minutes, gathers the various statistics, and writes them on the disk for later off-line printing. The printouts also contain a rough plotting device that assists the interpretation of the statistics. Samples of the type of information gathered in this way are the following:

1. percent run time for all users except background,
2. number of users logged in,
3. average number of commands per minute initiated by consoles and programs,
4. percent of time spent swapping users,
5. percent of time lost waiting for disk while users were reading,
6. number of files read per minute by commands and user programs,
7. number of files written per minute by commands and user programs,
8. number of words read per minute by commands and user programs,

9. number of words written per minute by commands and user programs,
10. average of final links of user programs loaded using LOAD command,
11. number of completed interactions per minute,
12. average number of users thinking,
13. average number of users waiting.

The programs for gathering information in this form are now being completed and polished. As soon as they are available in final form, the various parameters of the system will be adjusted and the resulting effects on the system and the users will be observed.

SCHOOL OF ENGINEERING

Educational Applications of Multiple-Access Computers

General Method for Network Synthesis

Command Programs for Time-Sharing

Computer-Aided Ship Design

Computer Evaluation of Cauchy-Type Integrals

<u>Educational Applications of Multiple-Access Computers</u> - Henry M. Paynter, Thomas B. Sheridan, Ronald C. Rosenberg, Roger A. Humphrey

This project develops techniques with which a learner may experience an important aspect of creative development: the repeated posing, testing, and refining of hypotheses in the light of experiment. With such techniques a user will eventually make the effective use for his own needs of the computational, simulative, and memory capabilities of a general-purpose computer; and, the organization of his experiences will be specific to the user's needs.

In a doctoral thesis problem, "Teaching Dynamic System Behavior by Machine", Ronald Rosenberg investigates how we may give and control the conceptual organization of a complex subject with respect to a user who is interacting with a computer. Project Entelechon, under the guidance of Prof. H. M. Paynter, develops ways to teach the planning, execution, and interpretation of many logical and physical experiments for information acquisition. The current doctoral thesis work of Marc Weinberger is i the automatic synthesis of complex systems which satisfy a variety of performance criteria. Underlying all these efforts is the question of how we may improve the creative learning environment so as to produce better results.

We have developed a language for establishing a desired learning environment of which conceptual organization will occur. The language is that of bond graphs, an energetically-derived graphical symbolism with applications to many engineering domains, such as mechanical translation and rotation, electrical networks, and fluid systems. The language has distinct advantages: a system description in bond graph notation becomes the user's concept of the system, a convenient coding of a system for storage and retrieval by both the user and the computer is now possible, and the graphical language itself suggests further insights and can form relationships with the physical world in terms of primitives and aggregates.

A dynamic systems laboratory in the form of a digital simulator (ENPORT 1) is now operational in the first generation. This system simulates the behavior of certain lumped-parameter parameter, passive, linear systems; it further permits the creation of active source elements in a unique way. We have also conducted a teaching experiment which indicates that at least some users will carry out a considerable amount of exploratory behavior in response to general problem statements.

In the normal learning environment, we shall expand the simulator to cope with all lumped, linear, passive systems; and, we shall introduce a nonlinear primitive element, probably in the form of a degenerate characteristic, such as a diode. We shall also develop, in accordance with current concepts of programmed instruction, a control system which will monitor and guide the behavior of a learner. With this system the machine will exert concurrent influence on the user's actions as well as provide data for the experimenter to analyse and evaluate. Finally, the user will have at least one opportunity to relate the dynamic system models he is developing to a physical manifestation. To keep adequate records, the user must manipulate the physical system entirely through the console.

A time-sharing system that will meet the above needs should cause no perceptible delay of the user's thought processes. A virtually immediate response to user input should occur. Furthermore, output rates of the computer system should match user information acquisition and, possibly, decision rates. Console input formats should be extremely flexible and tolerant of user error; they should put the burden for interpretation on the machine, principally by means of suitable software. As a part of the CAD Project and Project Entelechon, we have developed some inexpensive equipment which, while having a rapid display of analog-generated signals, does not require excessive computer time.

ENTELECHON ZERO is an analog computer capable of automated setup and manipulation through a digital computer. The analog computer is equivalent to four K5-U's. For the sake of simplicity, the outputs will be semipermanently wired as a matrix array to the inputs. The computer will use eight solid-state operational amplifiers, with four running as inverters to provide positive and negative bus systems for the inputs. It now uses twelve K2-W operational amplifiers.

The input transconductor blocks are the unique part of the analog computer. Each block consists of nine resistors with binary conductance values connected in parallel from the (+/-) input bus to the operational amplifier input. Each transconductor requires a ten-bit binary code, including sign, for a setup that provides nominal gain settings from -15 to +15 in steps of about 0.031. The analog computer uses twenty memory blocks, sixteen for parameter setting and four for initial conditions.

The digital computer we are using in the development of ENTELECHON ZERO is an IBM 7094, operating in the Compatible Time-Sharing System. This system sends information in an expanded teletype code to modified teletype machines which are the input-output stations for the system. By using the signals directly as they come over the phone lines, ENTELECHON ZERO takes advantage of existing high-speed computer buffering equipment. The software system of the 7094 also provides an extremely flexible backup for ENTELECHON ZERO.

General Method for Network Synthesis - Marc R. Weinberger

This research work is the development of a general method for network synthesis. Though it uses the language of electrical networks, the method applies also to mechanical or structural circuits. Besides the usual linear elements, it includes a few basic nonlinear parts and, thus, has a wide variety of possible applications. In contrast with the classical procedures, time-domain synthesis is here directly feasible. We do not employ the devices of classical synthesis, but achieve more general results by viewing the problem as a search and, then, by applying random search procedures. We may define a flexible criterion of goodness, according to the wishes of each particular user, by weighing not only purely technical performance but also cost factors such as size dissipation. In the search for a network the designer gains experience which helps him alter his strategies. He may also vary the criterion in the process according to whether some of its factors become more or less important in the narrow later stages of the search. For this method, a direct and easy contract between user and computer, such as the one established by time-sharing, is necessary. Though we may make several or even all of the steps automatic, we should not prevent the user from overriding this automatic portion at any time and then concentrating the search according to the experience he has gained, and thus from speeding up his convergence on a design.

During the reported period, we investigated the possibility of and the general methods for realizing our objectives from a theoretical point of view and, then, undertook a small problem of designing a delay circuit. From this problem came good results, and it provided information on the speed of convergence of the search, its need for adaptation of criterion and of strategies, and the point of its cessation.

We are now working on the problem of a coupling network for a pulse amplifier. This problem exhibits most of the features stated above, but is restricted to linear cases. It enumerates structures and assigns probabilities to them, has a general criterion with variable weighing factors as the search progresses, and has a time behavior analysis. In the coming months we shall implement the general problem outlined in the first part, particularly with regard to the ease of modelling the nonlinear parts.

<u>Command Programs for Time-Sharing</u> - Stephen L. Strong and Roy Kaplow

The objective of this program is the development of a system of programs which will enable a user with a vocabulary of ten or twenty commands to solve nonrepetitive computational problems on the computer via the time-sharing system. After a user initiates a command program, he communicates with the machine on a question-and-answer basis. There are, however, inherent short cuts for the experienced user who wishes to bypass interrogation by the system.

This system will be useful both in teaching and in graduate and staff research. In teaching, the system allows the solution of nontrivial class and homework problems of an experimental or theoretical nature, and yet does not require the students to learn computer programming. In graduate and staff research, it eliminates the programming for occasional or trial applications which is inefficient for both man and machine. In addition, it furnishes those advantages inherent in direct contact between user and computer.

We began our project during the period covered by our report and devoted considerable effort to establishing the ground rules for the system. We have already written subroutines for format-free input and subroutines for communication between programs and disk; we are now writing the system in a modular form, in which each command program is chained to by the main program, and to which we may add new commands as we desire them. The system is now able to evaluate any FORTRAN arithmetic statement involving constants, vectors, library functions, and ordinary arithmetic operations.

A simple example best illustrates the present development of the system. Given some data which represents a potential function U(r), we require the derivation of a density function from the equation:

DENS(R) = 4 R(exp(-U(r)/KT) - 1.0)

We use the following procedure:

1. We initialize the system.
2. The system indicates that the previous operation is completed and requests a new command.
3. We ask the system to print the input data.
4. Since the data is not on the disk with the name specified, the system asks if there is a pseudonym. If the data were loaded on the disk during disk edit, it would have as its secondary name the format of the data, i.e., DATA 10F7.3.
5. Since we have not previously loaded it on the disk, the data will be typed in now.
6. The data is typed in arbitrary format, with each data point separated by a space.
7. Now that it has the necessary information to complete the print command, the system prints the data, U(r), as requested.
8. The equation is now given as a command.
9. Since the constant pi is not on the disk, the system requests it. After this request the system knows that pi = 3.14159.
10. We now print the answer using the short form of the print command.

The system commands for integration with fixed or variable limits, for Fourier transforms, and for differentiation, have been outlined. Since the system became operational only in June, it will require some operational experience by others to ascertain its weaknesses and operational characteristics.

Computer-Aided Ship Design - M. L. Hamilton and A. D. Weiss

The goal of this project is a computer-aided system for the preliminary design of naval ships that assists and enhances the naval architect's work in the design process. A system of fluent communications between man and machine, which we hope to achieve with a design console that consists of a typewriter and a display scope, is indispensable to our goal.

The lag time between a decision to fix certain design parameters and the gathering of sufficient information for the evaluation of this decision is the most important factor in overall design time. The ability to make a decision and to immediately see the effects of that decision on the design is an invaluable aid in the preliminary design of ships due to bounds set by minimum weight, maximum volume, and proper stability. This triad of weight, volume, and stability characteristics results from a combination of hull parameters and hull shape, the arrangement of spaces, and the type of ship selected in the design cycle. We have programmed known relationships among these variables, and used graphical input/output devices for the description of hull surfaces and for the arrangement of space. This is the major portion of the computer-aided design process.

Our progress to date has been in two areas: (1) the evaluation of weight and powering requirements for destroyer types and (2) hull surface description and associated properties. In the first, we have developed a program which iteratively solves for ship displacement and shaft horsepower from a choice of destroyer hull parameters. In the second, we have worked on the generation of hull surfaces analytically with the technique of Professor Steven A. Coons that uses parametric curvilinear coordinates. We have designed and displayed such surfaces on the KLUDGE equipment at Project MAC. We have also worked on the evaluation of surface area and enclosed volume properties of such surfaces. This work on surface generation and display may form the basis for a later system which uses SKETCHPAD III for graphical communication between man and machine.

We have further developments to make in the hull design process, in the presentation of hull sections for purposes of arrangement, and in the adaptation of the computer aids for easy use by the actual designer. A series of parametric studies, on a half-completed package of such a computer-aided design cycle, may also be valuable to our work.

<u>Computer Evaluation of Cauchy-Type Integrals</u> -
Paul A. Wieselman

A unique method for the solution of the general plane stress o, strain problem is that of the Russian mathematician, Mushkhelishvili. This method transforms the biharmonic equation associated with the Airy stress function into a special combination of two analytic, and hence harmonic, functions of a complex variable. The problem is an evaluation of Cauchy-type integrals, for the solution interior to a complex domain or region that is specified by given boundary conditions, such as the loading of a stressed part.

The standard procedure for the evaluation of any Cauchy-type integral for regions bounded by complicated curves is the conformal mapping of such a region into a mathematically more simple region. This mapping is sometimes more difficult than the original problem. The present study seeks a numerical technique for the mapping onto the unit circle, either of a general region that is described in terms of the coordinates of points and smoothed by curve fitting or of a polygonal region of straight lines. The value of such a mapping in many other types of problems besides those of analysis is evident.

Our past work with the Project MAC facilities was the evaluation of some very-high degree, but approximate, complex, polynomial mapping-functions, whose coefficients we obtained analytically from the Schwarz-Christoffel polygonal-mapping formula. We shall soon complete a numerical minimizing technique that automatically evaluates the polynomial coefficients from the input data of the boundary description. We shall then apply the completely-automated mapping-technique to the stress analysis problem and evaluate its accuracy.

CIVIL ENGINEERING DEPARTMENT

A Preliminary Study of On-Line Structural Design - Gerald M. Sturman

Our goal is the use of the CTSS to aid in the design and optimization of civil engineering structures. The use of high-speed computers has not altered the trial-and-error nature of the process of structural design, because the analysis of an indeterminate structure, the most common type, depends on the properties of each of the structural elements. Thus, unless we know these elements, we cannot analyze the structure; and, to design a structure, we must intuitively choose the properties of the elements, analyze redesign them, and then further analyze, and so on. This process continues until a satisfactory design is obtained. A regular batch-process computer operation delays by several hours the results of each analysis. The design process has thus an artificial prolongation, and the designer has no real dialogue with the computer. We have been experimenting with an on-line structural design process that uses two existing programs written for the M.I.T. CTSS: the STRESS program for structural analysis, and the linear programming routine MFOR, developed by the RAND Corporation for structural design.

Intuitive knowledge of the behavior of structural systems, derived from the experience and training of the designer, guides his choice of a preliminary structure. Time-sharing may help facilitate this choice, since it makes many of the calculations necessary for a choice of elements in a preliminary structure on-line and partially automates them. The STRESS program analyzes the chosen preliminary structure. This analysis may be for either one system of loads on it or, in the general case, a variety of load systems. At this point the designer sees which load systems are critical and which may be ignored. Subsequent trials are more efficient, and the designer saves time by observing the behavior of the structure as he repeats the design cycle with the help of MFOR. Our current studies include the choice of a preliminary design, the increased ease in the switching of control from STRESS to MFOR, and more efficient methods of redesign between the STRESS and MFOR cycles.

<u>STRESS: Structural Engineering Systems Solver</u> - J. Melvin Biggs, Robert D. Logcher, S. Finkelstein, S. P. Mauch, Richard V. Goodman, Salvatore G. Mazzotta, and John R. Roy

STRESS is a problem-oriented language and processor for structural engineering that is intended as a design tool suitable for design problems and for an easy application of the computer to appropriate parts of the design process. It operates on the dissected components of the structural design process, so that the engineer may easily manipulate them to suit each desired application. The success of STRESS has been a step in the improvement of man-machine communication in structural engineering.

The number of variables in a design process is generally very great; and, in addition, the interdependence of many variables is nonlinear in even the simplest design. Traditionally, the design process is separated into components in such a way that only a small group of variables are in nonlinear relations, and the process itself is an iteration upon its components. We use the linear relations in an elastic analysis.

The number of equations required for analysis is a linear function of the topological size and a function of the type of structure. The amount of data required for, and resulting from the analysis varies approximately with the square of the number of equations. Due to the extensiveness of the calculations required, we have undertaken few analyses during design; however, we have devised elaborate approximate and special methods to obtain some semblance of analysis with hand calculation. We have used the computer extensively for structural analysis, but always in special cases, two of which illustrate the contrast between need and past use.

We have extensive computer programs for the analysis of very large or special complex structures such as radar antennae. Generally, these programs are not easily usable, alterable, or expandable; and they display a distinct remoteness from the engineering process. The vast majority of programs are for relatively small structures of special types, but in many cases policy forces the engineer to use them even when their limitations curtail his inventiveness.

For the development of a general design tool, we began in linear elastic analysis. This embodied a well-formulated but necessary component of the design process and had many characteristics of the whole, particularly the form and extent of both the data and the program. We have tried to provide a language for easy communication, for automatic data handling, for programming without direct concern for storage allocation, and for the most general capacities for flexibility, all within the framework of an easily expandable system.

The present language, described in the <u>STRESS-User's Manual</u>, attempts to simulate the engineer's natural language in order to maintain his association with the problem. The processor is a problem-oriented language, not only because of the free field and language form of the input that embodies such phases as MEMBER 23 RELEASES END FORCE X Y MOMENT Z, but also because of its generality of form and operation and its treatment of the components of the design process. STRESS is a communication tool: in input-error correcting or problem-specification debugging, in searching data for pertinent results, and in design evolution by successive modification of a problem.

Efficient use of both man and machine is based on the type, form, and method of input-output. The present form of I/O in STRESS is efficient within the desired limits of generality. Input, involving more than a few lines of data from a console to a large translation program, is slow and inefficient for a user and requires large amounts of swap time. Input to the disk, with the small monitor routine, is rapid and efficient. Disk input to a program, is also efficient, but the detection of an error forces an interruption in it.

Both user and program must be able to operate on the inputting process. The user types a statement, READ FROM FILE SAMPLE TRUSS, to continue the input process from previously-prepared input data. Error detection prints an appropriate message, and then returns control to the console for minor problem modification. The user still has the option of typing input from the console to the program. The machine can prepare output fast enough to fill its output buffer more rapidly than its economical computation rate for significant memory usage. Efficiency is then low and response poor.

We have altered the STRESS processor to place all output greater than a few lines onto disk files and, if we desire, to print these files with a very small program. The printing, linked to the program blocks with the command-chaining feature, differs from the block with its absence of common storage. Finally, the separation and size restrictions placed on console output programs, and the use of the disk as an integral component of significant I/O programs, increase the efficiency and speed by at least an order of magnitude.

In summary, the effort of the past six months has resulted in an operational STRESS via CTSS. We are now gaining experience in applying the system to practical problems in engineering design. Additions to the output capacity are desired, to decrease human response time in the required design decisions.

Transportation and Highway Systems -

John A. Suhrbier, Dale Gladding, R. Tepper, D. Moran

Introduction

This is a summary of time-sharing related research activity in the transportation and highway systems group of the Department of Civil Engineering. The personnel within this group are supported in part by the Massachusetts Department of Public Roads and in part by the Inter-American Program in Civil Engineering.

The objective of this research is the development of an integrated system of computer routines, computer and computer-related hardware, and engineering procedures which can solve the problems associated with the location and design of a highway transportation network. Because of the complicated relationship between the cost of a facility and its topography, these problems may be difficult. Such an integrated system is, therefore, necessary to maximize the benefits of investments both in the highway facility itself and in the engineering effort expended in the choice of the location. The research goals during the reported period were: the assembly of existing operable computer programs into a prototype route location system, and the further development of computer routines to aid engineers in selecting and designing feasible horizontal and vertical alignments.

The Route Location Problem

Location engineering makes decisions by means of operations upon information in order to produce a selected course of action. These operations are the identification of goals and decision criteria, the search for a set of alternatives, the prediction of their consequences according to some scheme, and the decision on the basis of these criteria either to accept an existing alternative or to continue the search process. An integrated route location system, such as the one discussed here, should therefore include computer routines not only for prediction and evaluation, but also for search and decision.

The route location problem is hierarchical in structure. Typical of the higher levels of analysis are the decisions as to what broad areas the system is to serve. Somewhat lower decision levels may be concerned with the planning of specific projects and with the preliminary location of individual links. The lowest levels of this hierarchy are represented by engineering design and the preparation of design plans and specifications. A route location system must include not only routines which are applicable within each level, but also procedures and techniques for controlling the progression between levels.

The man and machine should be able to work together as a single problem-solving system. This interaction determines strongly the language in which the engineer describes his problem, the means with which he specifies the type of solution required, and the form in which he displays the solution. Decisions concerned with the location of transport routes are usually not clear cut; therefore, the engineer must be able to examine this uncertainty through such techniques as sensitivity analysis, statistical decision theory, and the monitoring of real world conditions.

Development of a Prototype Route Location System

We have designed and made operational a protytype route location system. Although the current system includes only the basic routines of the Digital Terrain Mode (DTM) Design System, the system structure is open ended so that we may easily incorporate routines based on the DTM Location System, the Vehicle Simulation and Operating Cost System, the Horizontal Alignment Selection Program, and the Vertical Alignment Selection Program. Embodied in the system are the concepts of a problem-oriented language. The input communication is in terms of program calls, such as EDIT TERRAIN, SIMULATE VEHICLE, DESIGN VOLUMES, and SELECT ALIGNMENT; input descriptions, which enable the engineer to input certain design conditions and a typical roadway cross section; alignment descriptors, which enable the engineer to describe and to modify the horizontal and vertical alignments; and output descriptors, which enable the engineer to specify the tabular and graphical output with a series of OUTPUT and PLOT statements. The alignment descriptors are an extremely important part of this prototype system. During the course of a single session at a time-sharing console, an engineer can not only examine a number of alternative alignments but also make numerous modifications to these alignments in an attempt to improve them.

Automatic Selection of Horizontal Alignments

Dale Gladding has formulated and developed a technique for the automatic selection of horizontal alignments. He adapted this work to the CTSS system, so that communication between man and machine could be improved within this individual routine, the Profile Selection-Evaluation System, to be described later, could be more easily used to evaluate the selected alignments, and that the routine could be incorporated into the prototype route location system.

He investigated several methods of alignment selection. The one chosen involves the determination of surfaces of potential vertical alignments by means of a mathematical smoothing technique. By examining the potential road surfaces, along with the original terrain model, he obtained a transportation cost function (or cost model) that involves both construction and user costs for the area considered. A least-path algorithm may search this cost function to determine the minimum cost path between two points, and this path and other feasible routes may be plotted automatically or manually on a map.

Profile Selection-Evaluation System

John Suhrbier developed the logic, methodology, and computer programs for an integrated-profile selection-evaluation system. The system calculates the horizontal geometry, generates and plots the ground profile, selects a tentative highway profile by means of a mathematical averaging technique and plots it over the ground profile, calculates the earthwork volumes associated with this profile, simulates the operation of vehicles over this alignment to determine operating and time costs, summarizes the results in the form of total and annual costs, and asks the engineer whether he would like to improve the selected profile. He based his calculations on the theory of the Digital Terrain Model and employed in them the routine and programs extant in the DTM Location System.

The highway profile must generally meet certain engineering restrictions, which are commonly associated with grades, sight distances, and control points. In addition to these engineering constraints, there is an economic principle to be satisfied: the vertical profile selected should minimize the sum of the initial construction cost and the continuing user cost. He wrote this program so that the selected profile satisfies these two types of restrictions.

An important and crucial part of the system is the feed-back loop in which the engineer can improve the profile selected by the machine. An engineer may gain a very good notion of how slight shifts in the vertical alignment influence both construction and user costs. Experience has shown that he may gain this intuitive notion after a few minutes of playing with the profile at a time-shared remote console.

An adaptation of the Profile Selection-Evaluation System to the CTSS system has improved communication between man and machine, especially in the feedback loop where interaction is now entirely in the English language. The use of the system thus became both easy and rapid in Mr. Gladding's experiments with the horizontal alignment selection program.

Further Time-Sharing Activity

In addition to the design and development of an ultimate route location system we plan the following work:

1. an incorporation of the Vehicle Simulation and Operating Cost System into the prototype system,

2. an investigation of ways in which the traffic assignment program, developed by the Transportation Demand Research Project in the Department of Civil Engineering, and the prototype route location system might be used together,

3. the further development of the horizontal and vertical alignment selection routines to improve their usefulness as engineering design aids,

4. the possible field testing by the Massachusetts Department of Public Works of the existing prototype system on one of their current location problems.

The Transportation Demand Project - William F. Johnson

The designing of an urban transportation network is a complex iterative process that requires much data and computation. Analytic techniques are not sufficiently well developed for the engineer to simplify the problem and still achieve reasonable results. He may use computers for particular stages of the design process but uncertain data and nonsystem oriented techniques preclude quick, one-step solutions. Optimization is mostly guesswork. Current techniques make evaluation of alternative networks difficult because of the work required to specify a network. Present design studies attempt the selection of alternatives and the evaluation of their relative merits in a limited amount of time with incomplete data.

The traffic assignment technique under development by the Highway Transportation Demand Research Project has several unique features. It simulates the loading of an urban transport network by specifying cost functions for the links of the network and by associating the demand for transport with travel between pairs of nodes. The cost of travel over a link increases as the number of trips taken over that link increases; the number of trips between two nodes decreases as the cost of travel between the two nodes increases; and, the equilibrium number of trips between two nodes is the number for which the cost and the willingness to pay balance one another. Equilibrium for the entire network is achieved when the equilibrium number of trips between each nodal pair is determinate. We select nodal pairs for assignment in random order, and we assign trips between nodes of the transport network in increments.

Our present research attempts to evaluate the effect of the random selection process and of the variations in increment size on the final network simulation. Our techniques are flexible enough so that we may evaluate many assumptions underlying present traffic assignment methods. We are also investigating the interrelationship of assignment methods, the machine configuration, and the character of the transport network planning problem.

Our research seeks out methods for making the digital computer more useful for transportation planning and design. It will include programs with which an engineer may easily specify and change a network. We shall also determine whether a slow speed input/output device, such as a time-sharing console, can handle large amounts of data and shall try to structure problems so that they highlight that information on exception which is necessary to the engineer for a quick understanding of the effects of network changes.

Highway Route Location - Marvin L. Manheim and James E. Burke

A number of computer programs perform analyses in different phases of highway route location: the DTM Design and Location Systems for evaluating locations in terms of earthwork and other construction costs, the Vehicle Simulation System for evaluating the user-cost effects of locations, algorithms for generating horizontal alignments and vertical profiles, and the Assignment program for predicting the distribution of traffic through a network.

We are now developing an integrated highway location system in which all these programs are available to the engineer. For us to integrate them into an efficient problem solving process, this variety of tools requires a scheme by which we could decide just which of these programs and their associated procedures to use at a particular stage in the solution of a highway location problem. The object of our research has been the development of such a scheme and its implementation in the form of a computer program.

The scheme represents the location process as the execution of a sequence of experiments which uses a particular set of programs and procedures. With Bayesian decision theory we have developed a model for computing, at any stage in the location process, the best experiment for the engineer to perform next. After the engineer supplies essential judgements in the form of subjective probability distributions, the computer program, GUIDE I, explores with a statistical decision-theory computation the implications of these judgements and then determines the experiment for which the total expected return is greatest.

The availability of CTSS enabled fairly rapid development and debugging of the program. Time-sharing was invaluable for explorations of the properties of the program, particularly of the dependency of the recommended action upon alternative forms of computational approximations, expressed in both parameter settings and the use of alternative subroutines, upon changes in the data of a problem, and upon constraints on the basic engineering process itself. The general logic is that of the tree exploration kind which necessitates heuristics to make computation practical. We developed a number of heuristics and conducted experiments with different combinations of them. The time-sharing mode allowed us to begin an experiment, to observe the results in progress, and to terminate the computations when we were satisfied with the combination of approximations. Thus we could try many different combinations and yet not use the computer time required for exhaustive execution of the computations. In addition, we developed a program with alternative levels of print-out detail at the console so that we might adjust the level of print-out according to the information we desired in a particular run.

Looking ahead to eventual practical utilization of Guide I in an integrated route location system, we foresee the following mode of operation. While the engineer executes a particular set of route location procedures in the foregound of the time-sharing system, Guide I works in the background to compute those location procedures the engineer will use next. Upon completion of the foreground procedures, the engineer calls Guide I into the foreground, examines its output, and recommends the next procedures to be used; then restarts Guide I with the new conditions, returns it to background, and finally begins execution of the new location procedures in the foreground. Time-sharing would thus allow the engineer to move easily between the engineering procedures themselves and those methods which aid him in choosing the engineering procedures.

<u>Traffic Flow Analysis</u> - Wayne M. Pecknold and Kent L. Groninger

Our initial objective was a working understanding of the techniques available with interaction between man and machine, of their use in the solution of traffic problems, and of their applicability to the problems of a broad spectrum of users. We developed programs for simulation and for statistical testing of traffic data which were composed of vehicle passage times at a stationary road point. We then transformed the data into histograms and, presuming a known interarrival distrubution, we performed on it goodness-of-fit tests for significance levels. These tests require a first approximation to some unknown parameters. After expending much effort to estimate them, we concluded that a graphical on-line plotter would greatly increase accuracy and reduce the time needed for calculations and that, in any event, visual aids would be useful in this type of problem.

In the second part we refined further these programs for traffic problems and, using queueing theory with a Markov process, constructed a probabilistic systems-engineering model of a signalized intersection. The dynamic control of traffic will require close interaction between man and machine, if we are to use computers for this control in real time. Therefore, we developed a model to determine the dynamic aspects of traffic flow rates and the conditions and periods of time for which they can be assumed stationary.

We wrote several subprograms for future use in transportation problems, for example, a program to handle the assignment problem which determines optimum allocation of traffic flows on a network and a generalized Poisson probability program. We also attempted jointly a linear programming approach to the synchronization of traffic signals. This attempt gave us the opportunity to test the usefulness of interaction between the users of a particular problem number common file.

Future objectives of work under the direction of Professor Bisbee are: first, the further development of traffic queueing problems and of solutions with emphasis on the efficacy of programs developed to process actual data supplied by the Port of New York Authority and, second, an investigation into other areas of traffic flow and of transportation problems by means of stochastic models built on the information gained from the queueing problems mentioned above.

Optimum Allocation of Traffic Flow - Alan M. Hershdorfer

Our problem in traffic flow is, with a given arterial street or freeway network and the traffic demands made on it, to analyze the effect of an increase of road capacity or of a creation of one way streets on the optimal flow pattern and on the total network travel time. The formal specification of the problem was as a nonlinear program soluble by a linear approximation. The revised Simplex algorithm, Share program RSMFOR, developed as a foreground time-sharing program, is suitable for time-sharing operation because it preserves the original problem matrix, whereas the standard Simplex method does not. The program user may change conveniently one or several problem parameters and then update the solution without again having to read in the unchanged data and to start the calculation from the beginning.

We have structured the time-sharing version of the linear program as a problem-oriented language with its own set of linear programming commands. We may control the program directly by means of the console or of pseudo-tape files. The program permits, all in core, the solution of relatively large linear programs: up to 511 rows, 2000 columns, and 7000 nonzero matrix entries. We are preparing a user's manual for the time-sharing linear program. We hope this program will be useful to all at Project MAC who deal with optimization problems. We designed the program to facilitate the interaction between man and machine in the problem-solving process, especially in those problems in which we must view the problem statement in the light of a previous problem solution. In particular, we hope that it might be integrated with STRESS.

The linear program associated with the optimal traffic flow problem has a block diagonal structure which allows a solution as a series of related subproblems by means of the Dansiz-Wolfe decomposition technique. We have share distributed to apply this technique (SMDASS-SMDCOMP) and are developing a time-sharing version of this package. When we have completed it, we shall have made available to time-sharing users a powerful tool for solving block diagonal linear programs of great size.

We are also considering the development of time-sharing versions of two nonlinear programming codes. The first, SCM3, allows piecewise linear constraints and objective function; the second, SDGP90, an arbitrary objective function with linear constraints. In conclusion, the study of a problem in network theory has led us to the development of a library of time-sharing mathematical programming codes. We expect this development to continue through the coming year.

Traffic Simulation Studies - Jay R. Walton

The unique features of a time-sharing system have caused my original goals to expand and my original methods to be modified in the following ways:

1. The general traffic simulation program, which I had envisioned as a single large subroutine with variations through use of parameters, is now taking the form of a GPSS-COGO language that allows each user to write his own simulation model. This change takes advantage of the ease of altering programs and avoids the long service times associated with large programs.

2. Whereas intermediate output was formerly ignored, it is of major interest. Through effective use of it the engineer can monitor simulation and control its course.

3. The anticipated use of simulation models has extended to their application in the determination of rational traffic control policies. With time-sharing, an engineer may simulate traffic under some initial control policy, watch and see what happens, adjust, and so on. In these cases he seeks a control policy that produces good flow conditions rather than the solution of a specific intersection or network problem.

4. The communication potential of time-sharing has emphasized the inadequacy of standard I/O devices in the realization of that potential. In batch processing, the user is relatively unaware of, and unconcerned with, the mode of

output of data. On time-sharing, however, he generally must watch as his answers are produced serially from left to right, line by line, digit by digit, and letter by letter. He does not want this serial production; he wants a block of data, a table, a matrix, or a complete listing of his program; and, he wants it quickly. Devices that can produce this kind of output are, in the order of increasing satisfaction to the user: a line printer, a page printer, and some device that converts blocks of binary output into an image on a TV screen. With the latter procedure, a user could look at his output, as he now looks at microfilm, by rolling the image forward or backward to the desired information.

Because of these revisions I have not made much progress towards completion of a general simulation model; instead, I have modified my previous work on one-way traffic to include some of the above features. At present the model represents movement along a one-way roadway. The parameters of the model are the length of roadway, distribution of entering times, velocities, following distances, and the simulated time. There are options that will print out queue length, delay measures, and positions of vehicles. Any of the parameters of the model may take on different values during a run.

In addition, I have attempted to apply the time-sharing version of GPSS to traffic simulation for specific models. They are the following: the intersection of Massachusetts Avenue and Memorial Drive, the intersection of St. Paul's Street and Beacon Street, a generalized "T" intersection of two roads of two lanes each, and a one lane, one-way roadway with signals.

Railway Engineering Systems - Norman W. Luttrell and S. R. Ditmeyer

Our objective has been the development of a large-scale real-time railway engineering computer system. The initial work consisted of programs to simulate the running of a railroad freight train over a track that was described by input data. The finished computer model of a rail vehicle will be quite comprehensive. Our main algorithm uses propagation through finite steps of distance. However, for long sections of track of constant gradient, the nonlinear differential equation of the train's motion is integrable and has solutions in the form of hyperbolic functions.

This single-step approach, although more complicated than the finite difference method, permits faster computation in cases which otherwise would require many finite steps. Our approach to this simulation problem includes a calculation of braking forces and train deceleration during the period of air brake application and subsequent release. This problem is quite involved due to both the slow rate of propagation of the braking action down the length of a long train and the variability of rates between different cars in the train.

Our next project, which comes in response to a suggestion from representatives of the New York Central Railroad, will be a study of the feasibility of an on-line query and reply system designed to handle those real-time adjustments in freight train schedules and connections that arise from delays to one of the trains. At first, core storage will contain the following information for each scheduled daily freight train on the railroad: train number, summarized schedule, and a shortened description of designation, as Cleveland, Pittsburgh and East, Buffalo, Syracuse and East. It will also contain information on the scheduled

12

connections to be made with other trains.

As delays are reported to the console operator, he types them in on-line. The proposed program will find the connections about to be missed and then calculate the total delay, which results from this missed connection, to all the cars in the train at their final designations. If the operator finds these delays unacceptable he will have the prerogative of making the connecting train wait for the delayed train. This wait will become a delay for the second train and will result in more calculations by the machine and more decisions by the operator. He may type in new delays at any time. In effect, then, human judgement still controls decisions, but with a better knowledge of the consequences of each decision than it has at present.

<u>Soil Engineering Problem-Oriented Language</u> - Robert L. Schiffman and Laurence N. Beckreck

SEPOL (Soil Engineering Problem-Oriented Language) is a program designed to implement the interaction between man and machine in handling problems which arise in soil engineering design and research. A main program of the language introduces the format mentioned below, initiates the values of variables, requests information on the type of problem to be analyzed, calls the appropriate subprogram to initiate the right analysis, offers the initiation of a new problem or a change of parameters on the previous problem, and then gives the new results.

The system is based on a hierarchy of calculations in a tree-like structure in which each subroutine analyzes a particular part. In this way, the desired results are compiled along the branches of the tree and filter back down to the main program. Each subroutine has its own direct communication facilities with the user. When he needs a parameter, the subroutine seeks a determination of it at some other point in the analysis and then uses the value found. If it finds no determination, the program calls for the value or calculates it from some other values.

The language includes several important methods of settlement analysis. Wherever certain options are not yet in the program, a subroutine prints out a statement to this effect. The storage of variables and of arrays for those values needed more commonly than by the subroutine is in the common area which we allocate by program. Thus, we avoid the restrictiveness of the FORTRAN DIMENSION statement.

Whenever the program offers the user options, as choices between methods of analysis or input, he specifies a choice with the floating-point number suggested by the program. He can use only floating-point numbers, one per line, for an input of numbers. An appropriate subroutine accepts the answers "yes" and "no" at certain points. The use of numbers to specify options simplifies both the writing and the use of the program. However, when we add the free input format that we have planned, we shall specify either the number or the option itself. This specification will require an input analyzer to determine the substance of the input statement. Although we have not yet determined the value of a free input format for this type of language we shall include a certain amount of it.

Intended modification will permit the user to break the question and answer sequence controlled by the program and to go into a manual mode. He will then be able either to specify with a limited set of commands extra output values, to return to a previous point in the communication, to change a value and continue, to call for a leave in an abbreviation mode of questions whose interactions he knows, or to demand an explanation of a particular method.

We shall add many other soil problems and methods of analysis from areas besides settlements such as stability analysis and seepage. The language we are writing can evolve with the changes and needs of soil engineering, since its hierarchical tree-like structure provides for easy additions of new methods.

Teaching Machine System - Daniel Roos

The teaching machine project includes both the design of an operational teaching machine system controlled by a time-shared computer and the use of this teaching machine to present course material that explains the compatible time-sharing system to interested members of the M.I.T. community. The teaching machine itself has three parts: the computer and computer program which controls the operation of the system and contains the necessary decision-making that the computer must perform, the remote console, which is the principal communicating device between the student and the computer, and a display device that contains the course material to be presented to the student. At present either a teletype or 1050 unit is the remote console, and a simple looseleaf binder with index tabs to denote sections serves as the display device.

The course material is divided into a series of concept blocks each of which contains a description of some aspect of time-sharing. The student sits at a remote console and reads an assigned concept block from the course material in the loose leaf book. He is then asked to perform some operation with the remote console. The computer receives, analyzes, and then comments on his answer; it also types the assignment of a new concept block. Since the teaching program contains branching, the assigned block depends on the answer the student has just typed. The next block may contain a review of the material in the last concept block, an explanation of any error the student made in his reply, or a presentation of new material. Each student follows a different path through the network of blocks.

The course material is designed so that the student will experience a typical time-sharing session. He is requested to perform actual time-sharing operations. At the conclusion of the course he has typed in, compiled, debugged, loaded, and run several programs. The student should not feel that he is taking a test when he is asked random questions, but rather that he is solving a problem step by step in the presence of a teacher who helps him when he goes astray. At the end of the course the students have a permanent document of a sample time-sharing run which can be used as a reference for other time-sharing runs. For this program to be effective, the student must freely construct his own answers. Although multiple-choice questions are more convenient, their educational benefit to the student is not so great.

The design of the computer program that contains the necessary decision making is as general as possible. Many of the routines thus can teach a completely different course in the same framework. All routines are in FAP. Since the program uses constructed response answers instead of the multiple-choice type, the majority of it deals with the testing of the student answer and the determination and specification of any error committed. Thus far we have written all the programs for the teaching machine system and are now debugging them. The course material has been written and reproduced with DITTO stencils.

Whenever we finish debugging and begin testing the system, we shall determine how effectively the system teaches time-sharing and how long, in terms of both machine and console time, the course lasts. If our teaching system investigation should prove encouraging, then we shall conduct research on display devices for course material, on a teaching system which a user could initiate at any time during a time-sharing session and from which he could request a presentation of one specific area of time-sharing, and on the introduction of probability and statistical decision theory into the teaching program.

<u>The COGO Language</u> - Daniel Roos, Ronald A. Walter, and Harold C. Frazier

During the past six months we have been improving COGO and adapting it for use with CTSS. In particular, we have developed a new improved time-sharing version of COGO, a dynamic memory allocator for use with COGO, and a new COGO graphical display system called COGO T-Square. We have have also released several COGO reports and formulated plans for improved versions of COGO. It is well suited for time-sharing because of the continual interaction between man and machine present in the system. We have designed and developed three versions that maximize this interaction and permit incremental problem solving.

Whenever the computer notices one of the fifty detectable errors during program execution, it prints an error message and transfers control to the engineer at the remote console so that he may immediately correct the error. He then resumes the execution of his COGO commands from the point at which the error caused suspension of the processing. The engineer may interrupt a run at any point to modify his program. This flexibility is of advantage in the solution of those large scale civil engineering design problems whose final solutions require successive approximations. Thus after each approximation, the engineer uses the results in his choice of further COGO commands.

We have introduced modes of putting commands in the COGO time-sharing version to facilitate communication between the engineer and the computer. The engineer chooses one of several input modes and changes the mode whenever he wishes. Sometimes he will have his COGO commands read from the disk: at other times he will type in commands from a remote console. His choice of the proper input mode influences the efficiency of his problem solution. The engineer specifies the command output he wants and its place of reception. Output is separated according to its relative importance: final results may need immediate examination at the remote console; some output is not directly related to the final results and may be stored on the disk for documentation; intermediate results that do not add to comprehension may be omitted altogether. With a proper choice of COGO output commands, the engineer may select the output he receives.

Initial COGO time-sharing experiments indicate a need for faster console response time to reduce the machine time charged to the user. The large core storage required by COGO, together with the relatively small amount of program execution between console commands, make up the almost negligible machine time of actual execution. Most of the machine-time cost to COGO users is the result of program swapping and of other time-sharing overhead.

Dynamic memory allocation-methods that minimize core storage requirements can improve the performance of COGO; thus, only essential program segments should go into core from the disk or drum during any given time interval. We have therefore developed for use with COGO a loader and storage allocator, whose preliminary tests have been encouraging. We may also adapt them to other programming systems to improve performance.

COGO is a constantly-inproving dynamic language, and we have already begun work to incorporate the following features:

1. simplified input format for time-sharing,
2. revised FAP executive routine,
3. new FORTRAN or SLIP executive routine,
4. new input-output options,
5. new command capabilities.

We have begun preliminary work on a COGO of the second generation that will serve as a refined design aid. The present COGO stores only point coordinates and specifies distances and angles: the new COGO would store entire geometric sections, such as curves, spirals, and horizontal and vertical alignment sections, all of which could be easily retrieved and used and modified and they could also be manipulated to a final design.

COGO, although it was designed primarily for civil engineering problems, has other applications, such as the calculation of complicated chemical bonding relationships and physical force polygons. To insure that COGO will be readily available, we are working to place COGO on a CTSS command level. Any user will then be able to obtain COGO merely by typing the CTSS command COGO.

<u>Programming Systems for Project Scheduling</u> - John R. Brach, William H. Linder, and J. Keller

We have developed two distinct programming systems for project scheduling. The first system, now used in CTSS, combines a number of subroutines under a single problem-oriented language. Commands in the the form of familiar terms, together with applicable parameters for each command, comprise the language. We have done much work to extend the power of the language through a technique of updating the data base that represents the project network. This feature controls the project during its execution and any subsequent warranted rescheduling of the remainder.

The second system, which consists of ten compatible and fully-integrated programming packages is also fully operational in the time-sharing mode. This system reflects a new approach to large-scale project planning, scheduling, monitoring, and updating that is largely possible through the time-sharing mode of operation. Using the above program packages in the time-sharing mode, we may reduce the computational aspects of project control to the multiplication of two numbers on a slide rule. Some important characteristics of this system are the following:

1. It is geared to the user not to the programmer and as such is highly responsive to individual need.

2. The time-sharing mode of operation made possible the development of a usable new linear programming routine for that planning and scheduling which gives intimate contact between the user and his computations. With this routine the user need no longer supply vast quantities of time cost data since the program requests only requisite information, to which the user may direct his attention and, thus, with less effort may supply better data.

3. A new system of tape writing statements and formats, so that the user edits a pseudo data tape on disk rather than a deck of data cards.

We are also concerned with the application of machine-aided techniques to building design. We undertook a preliminary study to investigate machine-oriented, linear programming techniques as applied to the design process, and we developed a method to assign an arbitrary number of activities to an equal number of spaces with a minumum of total interaction. Development of our programming system will continue, particularly that of dynamic system operation by means of a special-purpose memory allocator. Work in building design will study in greater detail the building process to encourage the synthesis of computer aids.

Unsteady Flow in Open Channels - Ronald T. McLaughlin, J. E. Dailey, Christian Kim, Frank E. Perkins

The study of flow phenomena in systems of open channels and the optimum design of such systems has been hampered by the difficult analysis of unsteady flow in open channels. In our current research we are investigating these phenomena and are developing methods of analysis and optimization. The efficiency and speed with which an engineer can get answers are important aspects of this problem. The first part of our research is the development of a computer program that studies the movement of waves and surges in a single open channel. The program derives from the method of characteristics, which is used to solve nonlinear partial differential equations for velocity and water depth as a function of time and distance along the channel.

J. E. Dailey has developed a mathematical model of unsteady flow in rectangular nonprismatic channels. He programmed, debugged, and tested it with CTSS, which was found to be efficient for the quick removal of errors in syntax and the simple parts of the logic. Because of time limitations at the available consoles, he could not try actual runs on foreground and thus determine how well an engineer at the console could study a specific problem.

Christian Kim has used general geometry to develop a mathematical model for unsteady flow in open channels. The experience of the first model made the programming for the second relatively easy, but the typing of input data became more important. He handled data more conveniently with the keypunch than with the console; therefore, he debugged the program on the departmental IBM 1620 and tested it with the MAC console.

RESEARCH LABORATORY OF ELECTRONICS

Dynamics of Beam Plasma Systems

Numerical Solution of Space Charge Wave-Propagation Constants

A Computer Display for Wave-Type Instabilities

Speech Analysis

<u>Dynamics of Beam Plasma Systems</u> - Abraham Bers

One of the most useful macroscopic descriptions of a plasma is the dispersion relation for small perturbations. This relation gives the interdependence of the complex frequencies and complex wave numbers that characterize the natural wave-type responses of the system. Dispersion relations, in their simplest form, are polynomials with complex coefficients, but they usually involve transcendental functions as well. The variety of plasma parameters and the complexity of the equations are such that a complete analysis of a general dispersion relation is usually impractical. In such cases, computations are useful only if one can home in on the desired result by a continuous use of the answers which the computer gives to a simpler set of problems. Such an interaction is necessary when one uses the dispersion relation to determine the stability of a plasma system.

Recently, our group has developed mathematical criteria that give a complete description and classification of stable and unstable waves in plasmas. These criteria involve an analysis of the motion and mapping of contours of the dispersion relation in both the complex frequency and complex wave-number planes. With the aid of the KLUDGE display system we have programmed these criteria so that the wave solutions of any dispersion relation can be analyzed; and, hence, we can determine the stability of the system. We can also determine the stability, or the mode of instability, of a plasma system as a function of its physical parameters. We already have a determination of stability for a waveguide system of an electron beam interacting with a hot plasma. This computer display system for the analysis of wave instabilities is unique and has been received with considerable attention in the field of plasma research.

We shall continue to perfect the computer analysis and display of the stability criteria, and we shall develop programs for displaying higher-order singularities, such as multiple-order poles, the meeting of several contours, branch-point singularities, and branch lines.

Numerical Solution of Space Charge Wave-Propagation Constants

- Harold Schneider

We have successfully written, and used on the CTSS, a program for the numerical solution of the space charge wave propagation constants on a hollow cylindrical electron stream inside a waveguide. While approximate techniques are available for the lowest roots of the system of equations given below, this program provides for the exact solution for the higher order modes.

We can easily find the space charge wave propagation constants if we know the wave numbers p and q_o. Both p and q_o are real positive numbers which satisfy the two simultaneous equations:

$$p^2 = q_o^2 \left[\frac{\beta_p^2}{(\beta_e - q_o^2 + k^2)^2} - 1 \right] \tag{1}$$

and

$$F(p, q_o) = 0, \tag{2}$$

where $F(p, q_o)$ contains Bessel, Neumann, modified Bessel, and modified Neumann functions of p and q_o and β_p, β_e, and k are known quantities. Figure 1 is a graphical representation of equation (1) and (2) in the (p, q_o) plane. We determine the roots by iterative trial and error. To determine the lowest root, we try an initial value of q_o slightly larger than and increase it until $F(p, q_o)$ changes sign. Still higher initial values will lead to the higher order roots.

At each guess for q_o, followed by the computation of p and F, the results are printed out. The operator may notice that the function F does not change significantly with increments in q and that these increments waste time. He then merely interrupts the program and sets a larger value for the increment in q_o. To find a higher-order mode, the operator need look at only the printed result for the lower mode of a starting value for q_o. The system is so designed that he may decide as the work progresses, according to results in front of him, whether or not he wishes to find any more higher-order modes. Finally, although too large increments in q_o may cause two roots to be skipped and yet not change the sign of F, the person who uses the program notices with a little experience that p has a much larger value than the one it had for similar roots in previous cases. He may again correct the situation by interrupting the program and by using smaller increments.

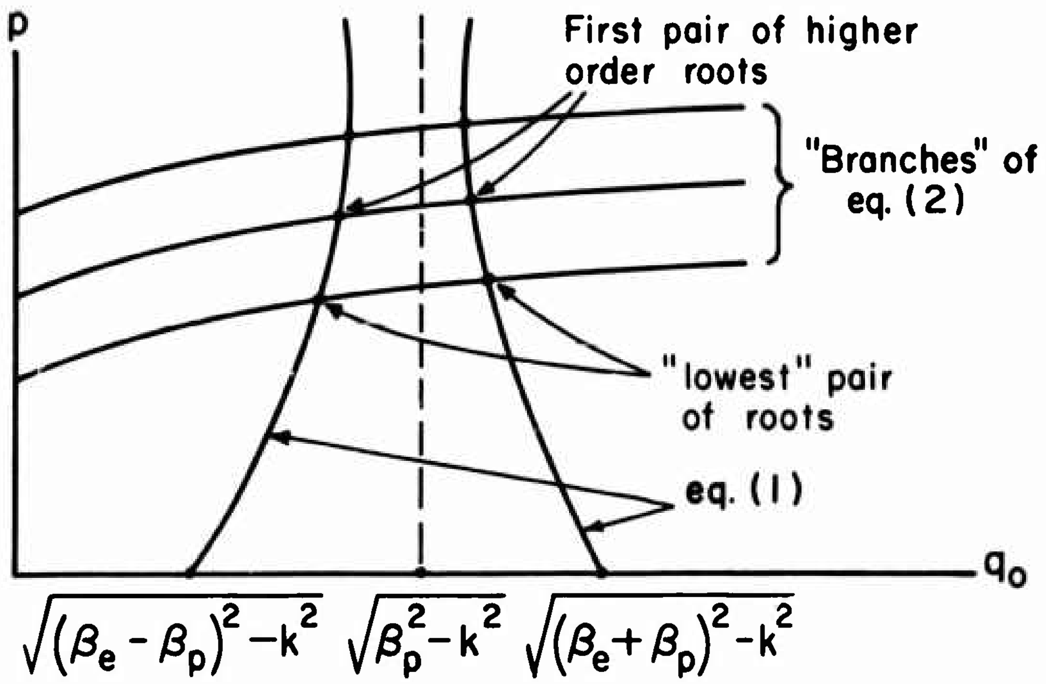

p
First pair of higher order roots
"Branches" of eq. (2)
"lowest" pair of roots
eq. (1)
q_o
$\sqrt{(\beta_e-\beta_p)^2-k^2}$
$\sqrt{\beta_p^2-k^2}$
$\sqrt{(\beta_e+\beta_p)^2-k^2}$

Figure 1. Constants of Space Charge Wave Propagation

<u>A Computer Display for Wave Type Instabilities</u> -
James D. Mills

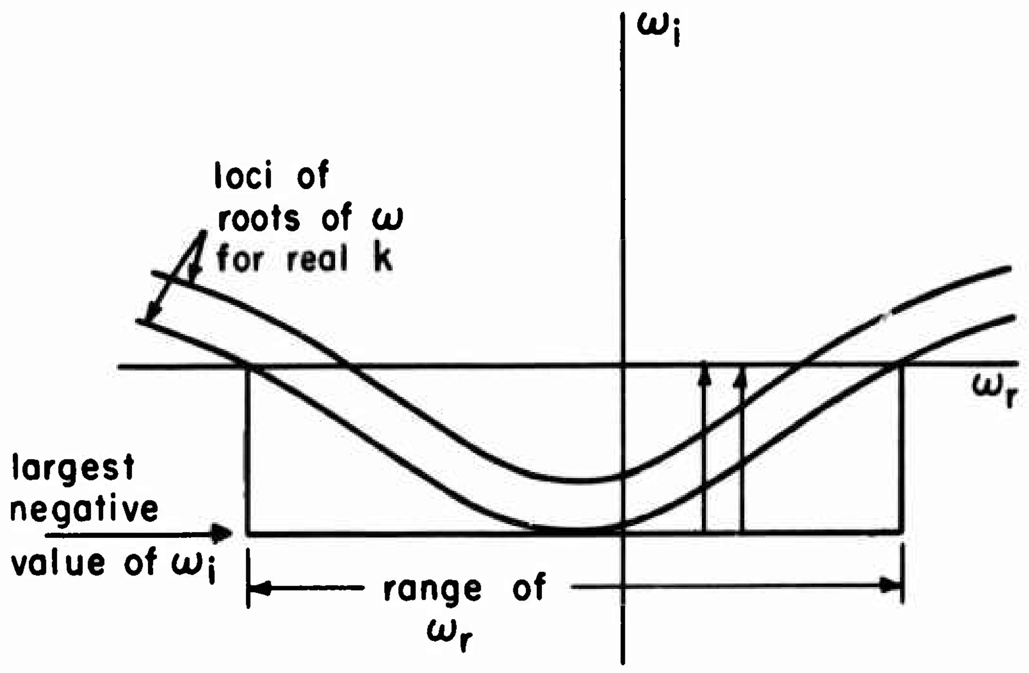
Bers and Briggs have recently developed mathematical criteria to determine and classify absolute instabilities, amplifying waves, evanescent waves, and the direction of signal flow in propagating waves. Their technique involves an examination of the solutions to the dispersion relation, $D(\omega,k) = 0$, for all wave solutions which are of the form $\exp(j\omega t)\exp(-jkz)$. This equation is a relation between the complex frequency, $\omega = \omega_r + j\omega_i$, and the complex wave number, $k = k_r + jk_i$. We solve for the roots of ω from the dispersion relation with real values of k and examine the solutions to determine the largest negative value of ω_i and the range of ω_r for which ω_i has negative values. These solutions outline the region of frequencies that are of interest for the determination of instability information, as shown in Figure 2. We then solve $D(\omega,k) = 0$ as a function of k with ω varying over this region of interest. In particular, we choose values of ω_r within the range and vary ω_i from a large negative value to the real axis. An examination of the loci of the roots in the complex k plane gives the desired stability information. Figure 3 shows these loci for two values of ω_r from Figure 2.

The dispersion relations for most systems of interest to us are of such a nature that a closed form solution is not possible. In general, we must use numerical methods of solution in the application of these criteria. We desire, therefore, to take advantage of the calculating speed of a digital computer. In the past, the length of time necessary for us to prepare programs, run them, plot and analyze the data, and rerun the program with adjusted parameters, has meant that the application of the stability criterion was a difficult and lengthy operation on a computer.

We propose, accordingly, to develop a computer program system that will accomplish two objectives. First, the program will permit the computer to directly use both equations and data in a format that is simple and familiar to a physicist or an engineer. Furthermore, the user will so be able to interact with the computer that he may change parameters and get new solutions in at most a few minutes. Second, the program will present results, on an on-line oscillographic display, of either the complex ω plane or the complex k plane and show the contours of the roots. The user will then see needed information at a glance and will not have to spend many tedious hours plotting roots on graph paper. This program will utilize the facilities of Project MAC for direct interaction between man and machine within the framework of the compatible time-sharing system.

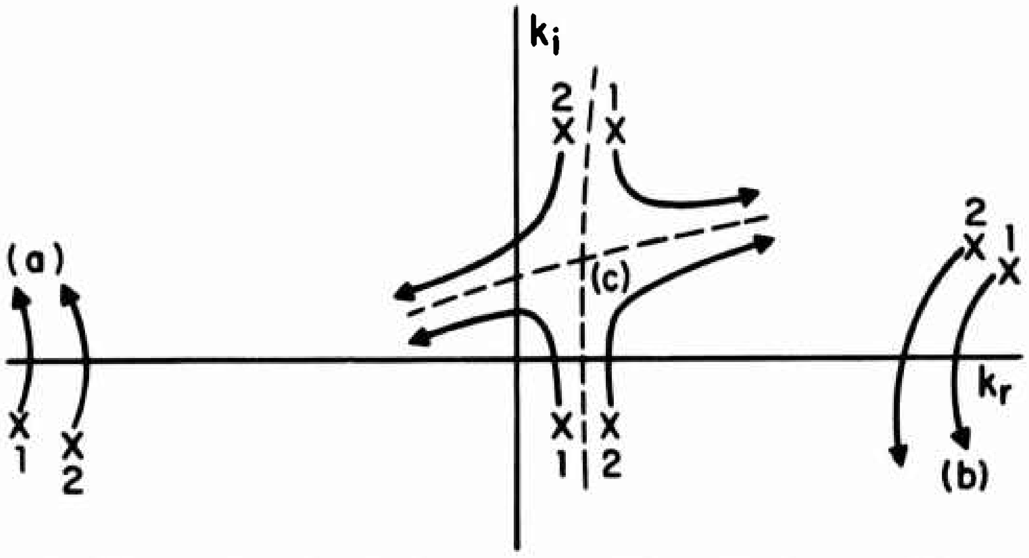
The program will be limited to solving dispersion relations that are polynomials in ω and k. Since most of the dispersion relations we are studying at present are polynomials, this limitation will simplify the program. For input, the user will type information pertinent to his problem into the computer in answer to questions printed out by the program on the teletypewriter. The program should accept the dispersion relation in a format very nearly like that which is familiar to the user; however, the symbol manipulation problems may present such an obstacle that more restrictive input formats will be necessary.

Subroutines available for solving polynomials with complex coefficients will facilitate calculation of the roots. When we have calculated the roots, we must, in order to plot the data, convert it into a format compatible with the display. We shall also use the program for a special study of multiple order roots of the dispersion relation. In particular, we shall investigate situations where roots coming from above and below the k_r axis meet to form multiple roots.



ω_i
loci of roots of ω for real k
ω_r
largest negative value of ω_i
range of ω_r

Figure 2. Complex ω Plane Showing Region of Interest.



(a) Convective instability (growing wave for $z>0$)
(b) Convective instability (growing wave for $z<0$)
(c) Absolute instability

Figure 3. Complex k Plane Showing Loci of Roots When ω varies as in Figure 2.

<u>Speech Analysis</u> - John M. Heinz

Using primarily the Electrical Engineering Department's TX-0 computer, we have been working on the development of programs for speech analysis that require interaction between computer and experimenter through oscilloscope, typewriter, light pen, and external equipment. During the period covered by this report, we have continued to work on this development at Project MAC, and we have initially tried to obtain similar programs for the MAC PDP-1 computer.

Several aspects of our present research are pertinent to Project MAC. First, we are developing equipment and computer programs for the efficient handling and presentation of speech spectral data obtained in on-line operation. Second, we are programming representations of models of speech production at the acoustic, articulatory, and other levels of description. These representations allow the experimenter to control the parameters of a given model so that he may both test and improve the model and obtain descriptions of speech signals in terms of the parameters of the model. Third, we are investigating various strategies for the automatic adjustment of model parameters that will so obtain matches between a model output and a speech signal under analysis as to provide procedures for automatic tracking of the parameters through continuous speech. A problem of less immediate interest to us is speech recognition by means of those parameters provided by present analysis systems.

Members of our group are also working on speech analysis through analog simulation of the vocal mechanism. We are currrently developing circuits directly controllable by a PDP-1 computer to provide a speech output for the computer. Program development is in progress for the specification of the control signals in a language more natural to the experimenter. We are also considering a digital simulation of the vocal tract.

The work done during the period using the MAC PDP-1 facilities is the recoding of a program, Type III, previously written for the TX-0 computer. This program provides:

1. a display of speech spectral data as a function of time,
2. the calculation and display of a speech spectrum, specified through typewriter adjustment of parameters, in an acoustic model of speech production, that is, pole-zero location of the vocal tract transfer function and sources,
3. a simulation of the filtering of the calculated spectrum with a filter bank of the same characteristics as that used to process the original speech signal,
4. a visual comparison of the calculated spectrum with the actual speech spectrum and difference curve,
5. a computation of various measures of error.

The recoding is now completed, and final debugging is in progress.

SCHOOL OF HUMANITIES AND SOCIAL SCIENCE

Social Systems Analysis

<u>Social Systems Analysis</u> - Ithiel D. Pool

A characteristic of the social sciences is that they are data-rich and theory-poor. The census volumes are a typical example. Present theory accounts but poorly for the numbers in the census volume. One simply goes out, collects them, and then uses them as empirical parameters. Similarly, in a recent election study, we used approximately one and a half million answers to questions by approximately 130,000 individuals to provide the empirical parameters for a model of voting behavior. The model itself was quite modest.

At present the social sciences must reach sophistication through empirical processes of measurement on a large scale. Naturally, computers have been useful to the researcher in his work with such massive data files. Up to now, however, work on large data bases has had to proceed in an inefficient fashion. To avoid the burdens of endless computation, the researcher has tended to narrow down the alternative hypotheses he tested. Even then, he has had to wait for his complete results to be returned from batch processing before he could evaluate each of these arbitrarily-selected alternatives. Our experiments with the MAC console seek ways in which large amounts of data can be conveniently stored, added to and subtracted from at will, and made available to quick experimental manipulation in order to permit the testing of hypotheses in a tree-like decision sequence, piece by piece.

An example from a current problem of Turkish survey data is the question of whether either literacy or mass media alone leads to the modernization of attitudes or whether the two must operate together by interaction. A large amount of survey data exists bearing in part upon these variables in the use of which, however, we must hold a number of other variables constant. The correlation of modernity of attitudes with each of the independent variables might provide a wrong answer if, for example, either mostly men are literate, or regional variations in fact account for the observed differences. In short, large numbers of interactions are possible which the design of a single batch run cannot wholly anticipate. The problem, in short, lies in the complex multivariate nature of social data together with the massive nature of social systems. We are working in three related directions which we might have described as three separate MAC projects, but find it more sensible to treat them as three parts of a single effort: the establishment of efficient survey bank and analysis programs, the Concom project for the simulation of the communications system of a nation, and the Crisiscom project for the simulation of the handling of information flows by decision makers.

Our Survey Bank Activities

At present, we are assembling approximately five hundred sample surveys from various foreign countries which include developing countries and also Communist areas. We are writing programs which store this information in packed form on a couple of tapes, which make any portion of it accessible, and which establish uniform formats in order to test questions of interest across a variety of surveys. Howard Rosenthal, Peter Ordeshook, John Nagle, Noel Morris, Tom Van Vleck, David Griffel, and Samuel Popkin are doing most of the programming for this effort.

The Concom Project

The Concom model accepts data from surveys and from a variety of other sources. This data describes the communications habits of the people of a country and the available sources of information and, then, represents the flow of this information within a national system so that we may estimate the extent of diffusion of information. We are working initially on the simulation of the communications system of the Soviet Union and of Communist China. We are now using the MAC time-sharing system to facilitate the development of this complex model in a fashion to make it compatible with the batch processing mode. We have also designed the system so that we may rapidly test with time-sharing and rapid feedback the effects of a change in the mode of communication on the diffusion of a message. This design would permit on-line experimental manipulation of communication alternatives. Samuel Popkin, Steven Sacks, Herbert Selesnick, John Kramer, John Nagle, and Peter Ordeshook are doing the main programming in this work.

The Crisiscom Project

Whereas the Comcom Project explores the diffusion of information through a communications system, the Crisiscom Project deals with that part of the received information which a decision maker will either accept or reject. The Crisiscom programs accept messages in an economical form and then permits their distortion, acceptance, rejection, forgetting, or remembering in ways determined by known patterns of human psychology. At the moment, the Crisiscom Project has two decision makers, J and K, that respond to a single flow of messages. We have programmed the Crisiscom simulation in the on-line version of SLIP and have designed it for mixed human-computer simulations.

The three efforts described above lend themselves in the future to the development of a general, social-simulation mechanism. They may assimilate empirical data about initial states of attitude and behavior, they may represent message flows changing these states, and they may take into account responses to these message flows. A general on-line social system simulator may not be practicable for some years, but it is the goal toward which we are working, and which the MAC system of time-sharing makes possible. Alan Kessler is doing the main programming on this project.

SLOAN SCHOOL OF MANAGEMENT

A General Theory of Human Problem Solving

Time-Sharing in Psychological Research

Management Applications
for the Development of Information Utilities

Marketing Model Construction

Computer Evaluation of a Technique of Stock Trading

Computer Job-Shop Simulation

A General Theory of Human Problem Solving - Peer O. Soelberg

Common to several behavioral sciences are situations in which people solve problems and make decisions. In order to advance artificial intelligence, engineers, who would normally develop better computing machinery and more sophisticated programming techniques, should study and evaluate these situations and the cognitive processes by which people resolve them. The objective of this project is the elaboration of a general theory of human problem solving. Current formulations, notably the "General Problem Solver" of Newell, Shaw, and Simon, deal with well structured decision situations where the problem to be solved, the rules of play, and the nature of the solution are defined explicitly with the initial presentation of the task , such as in chess, logic theory, and certain parlor games. However, most of the interesting problems in scientific research or in everyday human situations do not usually come equipped with either well structured definitions of the issues by which we decide or with the legal operators with which we reduce differences.

Part of this project is a study of how people solve a variety of poorly structured decision situations in their own environments; and, it should yield ideas and working data for the design of machine programs to accomplish similar tasks automatically. A recent version of our theory incorporates features of problem definition, strategy elaboration and selection, planning, learning, generalization, and search control that we have yet to find in corresponding special purpose subroutines.

During 1963/1964 we collected and analyzed behavioral data for two separate problem situations. The first was an experimental business environment simulation wherein a number of subjects interacted with a determinate, variably complex, and real-time computer program. Neither the initial definition of the problem nor the experimenter's explanation of the systems environment provided the subject with an operational means for resolving the task presented to him. By collecting on-line verbal protocols of subject-machine interactions in 462 sessions, we observed how different problem solvers researched their environments, learned to understand them better, defined subproblems that they wanted to attack, developed new operators and decision rules for dealing with the problems, formulated and tested hypotheses, and finally adapted their problem-solving strategies to changing information feedback.

The second problem situation used was a real-world decision situation. Through intensive interviews and periodic questionnaires over periods of two to four months during the spring of 1964, we studied M.I.T. graduate students who were deciding career employment and then tried to discover their information-processing strategies and methods for dealing with an unfamiliar, though real and critical problem.

These findings help us to remove from our interpretations of laboratory experiments those data biases which arise from the artificialities of environments simulated for experiments. Data from other field studies have already provided us with intriguing concepts to explore for a synthesis of the next version of the general theory.

The project has made and will continue to make the following uses of the Project MAC computational facilities:

1. on-line administration of laboratory experiments with a series of simulated problem environments,
2. interactive analysis of the experimental results,
3. synthesis and analysis of the subroutines making up the general theory of problem solving,
4. experimental simulation of indivual subjects, and

5. generation of statistical distributions for normalizing experimental and theoretical outputs.

In all but the last use, direct access to time-sharing consoles eliminates the inefficient and time-consuming use of independent machines. Since intermittent periods of data collection, analysis, and theoretical syntheses occupy much of our research time, our needs will peak sharply and unpredictably. We do not expect to be among the constant users of Project MAC's facilities in terms of calendar time.

Time-Sharing in Psychological Research - James R. Miller

I have been working on the adaptation of time-sharing to psychological research, particularly in human decision making, and on the development of instant statistical routines, particularly for the solution of common behavioral science problems. With the communication between man and machine and the flexible response obtainable from a time-sharing system, I am going to to collect and analyze data on human decision making. Via the console, progams that I have written administer various standard psychological tests to human subjects, play an economic bidding game with these same subjects, whose bidding behavior permits inferences about decision-making processes, and combine and analyze the results of both the psychological tests and the bidding game for each subject tested. These programs gather and analyze raw data for each individual subject, but at the present time no program combines these intermediate results into summary statistics on all the subjects tested. Future work on this project will develop a summary routine and will improve the communication process between the computer and the human subjects.

I am developing instant statistical routines to achieve a simple and swift procedure for the testing of various hypotheses commom to the behavioral sciences. I have written working programs for simple and multiple regression, for multiple correlation, for rank-order correlation, and for contingency and independence. Sloan School of Management personnel who have used these programs extensively over the past few months have felt that the time-sharing console is invaluable for the solution of problems of intermediate magnitude. If a problem is very small, a desk calculator or slide rule, together with published statistical tables, easily solve it; if it is very large or involves a great deal of input-output, batch processing is preferable; but if the problem involves only a few runs and a small amount of data for console input-output, time-sharing is preferable, especially in those situations in which a preliminary hypothesis requires an immediate, yet fairly crude test. Herein lies the greatest potential of time-sharing as a statistical tool.

During the next few months I shall improve existing statistical routines and create new ones. I have also requested the following additional routines: partial correlation, analysis of variance, factor analysis, binomial and multinomial tests, and other nonparametric techniques.

<u>Management Applications for the Development of Information Utilities</u> - Martin Greenberger

This is a summary of work done by M. Greenberger, M. Jones, M. Wantman, N. Patel, S. Whitelaw, and R. Welsh. The general objective of this group is the design, simulation, and implementation of broad management applications for the development of information utilities. The term, information utility, denotes a commercial realization of the widely

2

distributed and accessible computer system that Project MAC seeks to perfect. Specific goals of the group include the formulation of methodologies for such on-line operations as model building, scheduling, planning, constructing real-time management systems, manipulating statistical information, and structuring complex decision procedures. We have programmed a trial system for general on-line operations and process structuring, called OPS-1, and then applied it to a variety of management functions.

Our main reason for building the OPS-1 system was our desire to have a versatile modular framework with which we could construct incremental simulation models, or hybrid systems with simulation elements. Large-scale simulation activity urgently requires a way of flexibly running, testing, and modifying a model which has begun to assume a form, however embryonic. As a first step, we have developed an on-line simulation system, OPSIM, within the OPS-1 framework. The study and use of several current simulation languages, principally GPSS and SIMSCRIPT preceded the work on OPSIM and has led to certain results of intrinsic interest.

We modified and augmented the General Purpose System Simulator (GPSS) for on-line operation. The user of on-line GPSS may construct, test, modify, and run directly from the console a previously constructed and tested model stored on the disk; he may interrupt a simulation, run any point and examine the state of the system in any detail; and he may make changes and then either continue or restart the simulation. On-line GPSS was available as a public command on March 15, 1964, but since then has had only limited use, due largely to the low priority and poor service accorded GPSS by the scheduling algorithm. Its few users have found on-line GPSS a convenient and natural way to simulate.

In the fall of 1963 we modified the SIMSCRIPT compiler to fit within the MIT FMS batch-processing system and spent the following two months trying to adapt SIMSCRIPT to run on-line. SIMSCRIPT, unlike GPSS, compiles into object programs that refer to absolute common locations incompatible with CTSS relocation of common storage. SIMSCRIPT also requires certain standard FMS information not available in CTSS. These difficulties would have necessitated substantial reprogramming, had we not dropped the project when we had a working version of an on-line GPSS and saw no immediate user demand for an on-line SIMSCRIPT.

We are continuing the simulation of round robin and CTSS priority scheduling algorithms that was begun in the Fall of 1963. We have programmed a more elaborate model in GPSS which takes into account the arrival of users at consoles and the acceptance or rejection of their requests for service. Runs comparing the two scheduling algorithms have led us to certain conclusions about operating characteristics. We made a mathematical analysis of the two scheduling algorithms using standard queueing models. Stan Dunten of the MIT Computation Center has helped us to write a series of programs that collect statistics on the operation of the Compatible Time-Sharing System. These statistics, which we shall use to provide input for the simulation models of the scheduling algorithms, also help us understand the day-to-day operation of the actual time-sharing system.

We have already programmed and tested a number of improvements to the OPS-1 system. Within the next few months, we shall decide whether to extend this list and proceed with a more sophisticated version of the system, OPS-2, or whether to limit the list to the most important modifications and incorporate these within OPS-1. Among these modifications are facilities for the dynamic allocation of operators, the symbolic reference to storage, and the consolidation of parameter input. Applications of OPS-1 to the design of a national credit exchange, a vector processor, and an automated stock market are continuing. We are arranging visits with specialists and other persons at the New York Stock Exchange to gain a better understanding of their functions and shall arrange visits with other persons in government and industry as the situation demands. We are also pursuing theoretical O/R studies of space and time allocation and operator design.

<u>Marketing Model Construction</u> - John D. Little

Our project constructs models for use in marketing systems. A marketing manager must assemble data to modify his conception or model of the market; then, he must manipulate this model to make marketing decisions and to motivate more data collection. The quantity of marketing data, the depth of analysis warrented, and the complexity of realistic models in marketing all require heavy computer use. The marketing objectives of our research have led us to investigate what data should be collected, what models should be built, and how decisions should be made from the models. The computer objectives caused us to investigate how the computer can best aid this process, particularly during model building. The success of the interaction between model builder and computer and, eventually, between manager and computer will probably have an important effect on the speed of acquisition and the quality of marketing knowledge.

We have entered an extra company in the MIT marketing game. To operate the company, we perform experiments in the market, use the results to build models, and then use the models to determine actions, design new experiments, and so forth. The model builder obtains an increment of new information in each time period and, at any time, he may reexamine and modify his models. Thus far our operation has been primarily by hand with pencil, paper, and slide rule. The time necessary to do the clerical work to test a new idea has been a limitation. Since the difficulty with ordinary computer use is that the time saved in arithmetic is likely to be lost in programming, we hope the interaction will be more efficient with the MAC system. We may accumulate data as acquired and may construct models more quickly and test them on all past data, rather than on a limited sample. We may thus use the knowledge gained from a test more quickly to modify the model.

Our next work will review the company operations done in the spring. It will determine possible improvements from a marketing point of view and start the investigation of the programming required to facilitate the model building process. We view the present problem as the development of techniques centered around the game. If the techniques are successful, we will advance in two directions: first we will introduce the methods into live marketing operations, and second we will have students in the marketing game course use the methods to run their companies.

<u>Computer Evaluation of a Technique of Stock Trading</u> - R. W. Spitz

The objective of this study is the evaluation of point and figure charting. Many investors and speculators use this charting technique in their investment decisions, but very few have ever rigorously tested this method. This study attempts to show the return on investment that one might realize by using point and figure charting. A point and figure chartist generally maintains that the interpretation of stock price charts is an art, not a science, and that an intuitive feelin· for chart action is important. This study will also attempt to show how a digital computer may accurately perform the construction and interpretation of a stock price chart.

At present, many different methods exist for the prediction of future stock prices, but almost no documentation is available on the fruitfulness of any given technique. We may consider methods of predictions as either fundamental or technical. The fundamental one entails the examination and evaluation of such factors as earnings, assets, sales, and quality of company management, all of which determine a true value for the stock. If this true value is higher than the current market price then the stock is a good buy. The technical one, on the other hand, considers only price, and sometimes volume, over time. By treating the market itself as the best source of information on a stock and by observing the action of price over time, it expects to predict future price trends.

Point and figure charting is a technical method now quite popular. For each stock the chartist examines, he maintains a chart of the price trends. By recognizing various patterns on the chart, he claims to know when to buy or sell the stock. Our study of point and figure charting seeks a series of computer programs that construct the point and figure chart from daily price data, look for buy and sell signals, trade according to some specific investment strategies, and then evaluate the results of this strategy. To test the method, we shall run the program for a random sample of stocks and then compare the rate of return obtained by point and figure charting with that of a simple buy-and-hold strategy.

We have already written and debugged the program that constructs the chart and are now writing the signal recognition and trading programs. The completion of this study will entail that of the programming mentioned above, with final evaluation runs for a random sample of stocks.

Computer Job-Shop Simulation - Donald C. Carroll and Theodore W. Schwenke

This project studies the design, test, and evaluation of an on-line simulation that solves problems of real-time production planning and control in a hypothetical computer job shop. This problem environment seems rich enough to permit extension of the resulting techniques to a large class of real-time management systems. It may also determine the feasibility of attempts by man and machine on very complex problems, particularly those which share with the job shop the characteristic of a combinatorial solution space. Because this problem has realizable dimensions, a joint product of this research should be an operationally useful production management system.

To solve this problem, we shall develop a simulation program that is compact, efficient, oriented for ease of experimentation, and sufficiently modular to permit easy modification. We shall also develop a set of interface programs to facilitate cooperation between man and machine on problem solving, and then we shall conduct experiments to solve those problems fundamentally inherent in the establishment of manning levels and the other parameters of a modeled shop. The communication mode will include the cathode-ray-tube display as well as the normal teletype.

Four major programming subprojects, some of which we may perform in parallel, are:

1. development of a simulation model with associated bookkeeping routines - in FAP language,
2. development of an interactive input generator and statistical analyzer - in MADTRAN language,
3. development and testing of automatic job sequencing rules - in FAP language with a printed output, and

4. the development of a man-machine interface for on-line simulation - in FAP and Electronic Systems Laboratory KKK language - that consists of static reports and displays, dynamic displays, and an information-retrieval system.

A discussion of the current progress and future schedule for these subprojects occurs below. The complete debugging and testing of the model will take place in subproject 3.

Upon completion of these programs, we shall conduct a series of experiments in cooperative problem solving. At this stage the user may undertake several actions. He may do either of the following: assist the automatic, or heuristic, decision-making system with difficult decisions by means of programmed pattern recognition of difficult cases; set the system, or shop and decision rule, parameters through simulated trials; make system adjustments required by the occurrence of stochastic catastrophic events as machine breakdowns; or attempt, by continued introspection and observation, to formalize his own decision protocol for programming. Goals of these experiments include discovery of efficient methods of search for problem solution, evaluation of the form and mode of information tranfer, and, fundamentally, experience in cooperation. In this, as well as in the fully automatic system study, the results of our experiments should be directly useful in operations management systems.

At this time subproject 2 is completed, subprojects 1 and 3 are more than halfway done, and subproject 4 is progressing. By September, we expect a thoroughly-tested model with useable display and inquiry features. Consequently, experiments in cooperative problem solving should start some time in the fall.

/ 2

SCHOOL OF SCIENCE

Vortex Studies

Programs for Physical Problems

Energy Levels of Fluorine-77

Use of CTSS in a Plasma Physics Experiment

Vortex Studies - Henry M. Stommel

The equations of inviscid hydrodynamics, even in only two dimensions, are nonlinear; they contain many surprises and are still much investigated by applied mathematicians, physicists, engineers, meteorologists, and oceanographers. Solutions of physical interest are obtainable with simplifications such as linearization or steady incompressibility inviscid approximations. Despite the successful study of an impressive range of problems, which include difficult stability problems, various simplified kinds of turbulence, and so on, much of the subject is inaccessible to analytical methods and, accordingly, has stimulated the development of numerical methods.

Two or three dimensional time-varying fields, however, are so expensive to explore arithmetically, and indeed are often beyond the capacities of even our largest machines, that we are investigating various simplifications in the arithmetic treatment of hydrodynamics. Some have experimented with representation by extremely coarse grids and by limited numbers of Fourier terms. We, on the other hand, have been playing with the notion of representing flow fields by a small number of discrete vortices and of computing their interactions numerically from the exact, inviscid hydrodynamical equations. The interactions of more than two discrete vortices are so complicated that they are beyond analytical methods, but their solution is arithmetically economical.

Since a program already exists for use with the PDP-1 and its associated scope, which is both flexible and adapted to easy interaction between man and machine, we may easily and naturally experiment with the interaction of various numbers of vortices of various strengths, as though they were physically realizable and unusually controllable quantities in an experimental laboratory tank. The program running in the computer acts like an animated blackboard in which the chalk configurations of points, initially set up by hand on the board, proceed to interact and move on their own accord.

The goal of our vortex calculations was originally a study of models of atmospheric and oceanic circulations, in which the vortices are crude idealizations of the familiar highs and lows, but we quickly discovered that the many-vortex problem has much inherent interest and is not a trivial abstraction. Although our original goal has proved more difficult to realize than we had anticipated, our work towards it engendered an interesting, if unexpected, by-product: we found that we had available a microcosm which students could explore in various pedagogically fruitful ways.

The stimulation of a student's exploration of phenomena is difficult, because many of these phenomena have been so completely worked over that new and simple problems do not exist. We may, however, instruct a student in the use of the vortex program and then ask him to explore some interesting phenomenon. If he tries to understand some specific problem, like the stability of certain configurations, which the PDP-1 has suggested to him through a preliminary play with random combinations of initial conditions, and if he can find limiting configurations whose trajectories are computable by analytical methods, then, by formulating a problem within reach of analytical tools from a large and complex, albeit limited and artificial, numerical microcosm, he will have direct experience of scientific discovery. Several students who have worked with the PDP-1 in this way have had a taste of what that means. Design variations of this vortex program would supply other interesting miniature worlds in which to experiment.

<u>Programs for Physical Problems</u> - Elizabeth J. Campbell, C. Marceau, and Martha M. Pennell

<u>Programs for the Non-Programmer</u>
Since most of the people for whom we work do not know how to program, we have developed for them a set of techniques that requires only the knowledge of simple commands, such as RESUME and LOADGO, and also the ability to write a simple FORTRAN statement. The chaining of commands on time-sharing inspired us to write programs into which the user could insert algebraic information. An example is an integration program that we wrote mainly for demonstrations. When the user types LOADGO INTEGR, this program types out information that tells him how to type his integrand and how to put it in a file. The program chains INPUT, MADTRAN, and LOADGO in such a way that it furnishes the user with a Simpson's rule integrating routine which asks him for the limits of the integral and an initial guess at the number of points to try.

A similar but more ambitious program is Quixot. It actually writes another program, Sancho, which together with a Share Routine finds the roots of a polynomial whose coefficients are expressed algebraically. Quixot asks the user for the name of the parameters and the degree of the polynomial. While giving him specific instructions, it asks him to type algebraic expressions for the coeffiecients. Finally, it writes, compiles, and runs a program to do this particular polynomial. Quixot does not depend on the input command program, but forms a temporary disk file from console input and combines it with two previously written files to form Sancho, which it then compiles and loads.

Several other programs query the user about his variables, mode of operation, and so on, before they give him an answer. One is Bessie, which supplies Bessel and Hankel functions of integral order and complex argument and gives the user a choice of function and of coordinate system; another is Epoly, which does a least-squares fit of a polynomial equation with the Gauss Elimination Method. The output is both numerical and graphical, and thus lets the user easily decide whether his choices are satisfactory for his needs.

Subroutines for Programmers

We have written some routines for general use: one plots a graph of indefinite length on the x-axis, with the width of the paper as the y-axis, and a companion routine takes any section of that graph and expands it. A subroutine which sets and resets the input level in a MAD or FAP program will be extended to work in FORTRAN.

Programs for Specific Problems

The M.I.T.-S.L.A.C. (Stanford Linear Accelerator Center) collaboration is planning the electron-scattering program for the two mile linear accelerator now being constructed at Stanford. An important problem to be solved in the design and construction of two large magnetic spectrometers of 8 and 20 BeV/C, that will be employed in pion-electron discrimination. We can discriminate by detecting the x-rays emitted as synchrotron radiation by electrons, but not by pions, in traversing the fields in the deflecting magnets. The expression for the emitted photon spectrum was calculated by J. Schwinger as an integral of an irregular Bessel function. In the many mathematical decisions subsequently made in a relatively short period time, the on-line facilities of CTSS facilitated sensible guesses and estimates of the parameters and choices in the calculations.

At the moment, I am beginning a large project, the writing of what Professor Fano refers to as a kit of tools for the physicist. This kit will contain a set of subroutines which performs various operations needed in the calculation of nuclear spectroscopy problems and also several small main programs that use various combinations of these subroutines to perform certain operations standard for the nuclear physicist. At present I have a program for bound-state wavefunctions, binding energy, and radial integrals in which the user has an option of finding the binding energy and the wavefunction or of getting, for the same potential, several bound state waves and integrating up to a product of four of them plus a factor of the nth power of the radius, r.

I have been writing this kit in FORTRAN, since these programs ought to be compatible with other University installations, most of which do not use MAD. The physicists will use these programs for exploratory purposes. Then, as in minimization problems in the fitting of observed data to the theory, they may assemble these same routines for production runs, wherever long runs may be necessary for an accurate fit. There are programs in existence which do many of the things I have in mind but which are too unwieldy. Most people want to use only a few of the many options provided. The use of these options forces them to answer questions which are not vital to their problem.

I propose that a collection of small particular main programs which share a common body of subroutines will be more useful than one general program which tries to do everything. Sample subroutines will handle computations, such as bound or continuuum wavefunctions, various potentials, including the standard ones, programmed with the option of reading in either numerical values or algebraic expressions, various radial integrals, Bessel functions, relativistic and nonrelativistic Coulomb functions, minimizing routines, and so on. Statements such as BOUND STATE RADIAL INTEGRALS, RADIAL INTEGRALS BOUND and CONTINUUM, and COULOMB will activate the main control programs in this kit of tools. We plan to hold meetings of the Theory Group of LNS to discuss the building up of these routines, to formulate what is needed, and to define the notation to be used.

<u>Energy Levels of Fluorine-17</u> - Barbara A. Johanson and Frederic J. Eppling

We have thus far prepared our program for time-sharing and have demonstrated our problem, the examination of the energy levels of F-17. To examine these levels, we determine the values of the phaseshifts, $\delta^{\pm}_{\ell}$, which best fit the eight experimental cross sections measured at eight scattering angles, Θ, and at several values of the energy, E. The criterion of fit is that $\sum_{i=1}^{8} \ell_i^2$ be a minimum, where l is a relationship between the experimental and calculated cross sections. In this minimization we use the optimum-gradient method, an adaptation of the method of steepest descent.

At the beginning of the problem we read in first the eight angles and other constants, then guesses for the seven phaseshifts that involve minimization over seven variables, and finally the energy and the corresponding eight experimental cross sections. The program is so set up that, upon the completion of the problem for a particular energy, the answers for the phaseshifts are the guesses for the phaseshifts for the next energy. With TRA instruction we need read in only the next energy and experimental cross sections and, if the answers for the phase shifts are not satisfactory, we can dispense with a reading and resume the problem from its termination. In either case, we can avoid reading in the thetas and other constants for every new problem.

There are two more advantages of time-sharing for our particular problem. First, for a resonance, a sharp variation of one of the phaseshifts over a small energy range, we find a first guess difficult to make with the values of the phaseshifts from the previous energy value; but, with time-sharing, we can more easily work on these guesses and decide what intervals of E are necessary. Second, since more than one set of phaseshifts fit the problem mathematically, but not physically, physicists may examine the phaseshifts one energy at a time and thus avoid the wrong branches of the phaseshifts versus energy curves. In the future we hope to examine data at energies ranging from 2.7 MEV to about 4.5 MEV.

<u>Use of CTSS in a Plasma Physics Experiment</u> -
David T. Llewellyn-Jones

An experimenter, who must often use a digital computer to process the raw data from a laboratory experiment before its results are meaningful to him, finds the time delay of this procedure both inconvenient and annoying. When an experiment that is already restricted in time is further burdened by this loss of rapid feedback between the experimenter and his apparatus, it can entail the repetition of a complex, lengthy, or costly experimental procedure before the experimenter can act in accordance with the results of one single run of the experiment.

An example of such an experiment is the investigation of the dielectric properties of an ionized gas by the technique of interferometric spectroscopy. The experimental setup includes a high-density gas discharge for an infrared detector and a liquid-helium system, both of which the experimenter can operate only for short times. The raw data, essentially an autocorrelation function, is the output of a Michelson interferometer. Since the meaningful variables are contained in the Fourier transform of the interferometer output, which a computer evaluates numerically, the use of a time-sharing system as part of the experimental setup is desirable. Our objective is an automatic system which feeds the raw data directly to the computer from a remote console and at the same time allows the extensive intervention described below.

Input to the program is a set of eight parameters which the operator manually types in for each run. The main block of experimental data follows these parameters. This data consists of a hundred or more three digit numbers, which in the absence of an automatic system the operator also types in manually. We wrote a program that receives this input from a remote console. At each stage during the input, the program gives the operator an option of either making a correction or proceeding. If he wishes to correct a parameter, he types the name of the parameter and its new value; he can correct one or more data points by typing the index number and the new value of the data point. Because he attends mainly to the performance of his apparatus rather than to the manipulation of the console, the experimenter should have ample scope for correction. With this capacity for correction, the system has an unusually high possibility of human error, particularly since laboratory experiments rarely function with one hundred percent reliability.

We have the output of the program printed as a list of numbers and we also incorporate a crude plotting routine in the program. This routine is exceedingly useful for rapid assessment of the validity of an experiment. When we have dealt with a few remaining technical problems in the computer program, we can proceed with the hardware problem of feeding the data automatically from a digital voltmeter to the remote console.

Studies on the Nuclear Many-Body Problem - Juris P. Svenne

We are developing a program to perform a Hartree-Fock calculation of the nuclear many-body problem. It assumes that the nuclear wavefunction is expanded in a finite set of N oscillator wave functions, ϕ_α, such that:

$$\phi_a = Z_\alpha C^a_\alpha \phi_\alpha \tag{1}$$

where a and α represent a complete set of quantum numbers for the real nuclear wavefunction and the oscillator wavefunction, respectively. After it has computed the matrix elements, $\langle \alpha\beta | N_A | \alpha\delta \rangle$, of a phenomenological two-body potential and summed the effective Hartree-Fock potential, defined as:

$$\langle \alpha | a | \beta \rangle = \sum_{f,\gamma,\delta} C^{f_N}_{\gamma} C^{f}_{\delta}, \langle \alpha\gamma | N_A | \beta\delta \rangle \tag{2}$$

over occupied states, f, it obtains the single particle energies, , from the diagonalization of the Hamiltonian matrix:

$$\sum_\beta [\langle \alpha | \frac{p^2}{2m} | \beta \rangle + \langle \alpha | a | \beta \rangle] C^\alpha_\beta = E_a C^a_\alpha \tag{3}$$

The problem consists of the initial choice of a trial set of coefficients, E_a substitution into (2) for the effective potential, μ, diagonalization of (3) for a new set of coefficients, C^a_α, substitution back into (2), and, finally, iteration in this manner up to reasonable convergence. We have no way of knowing beforehand the number, N, of coefficients needed in the expansion (1) or the number of iterations required. Thus, if we can adjust these numbers directly from the CTSS console, we may maintain close control over the progress of the problem.

COMPUTER SYSTEM RESEARCH

System Requirements for Time-Sharing

Research on the 7094 Time-Sharing System

Research on Systems Programming Languages

System Requirements for Time-Sharing - Fernando J. Corbató

Most contemporary computers can form a general-purpose time-sharing system, especially after they have undergone modifications. Already the IBM 7094, the DEC PDP-1, and the Q-32 computer are fully time-shared; the CDC G21, the Johnniac, and the IBM 7040 participate in somewhat more limited forms of time-sharing; and, in the future, the GE 215 and the DEC PDP-6 may also be time-shared.

None of the existent computers are well designed for time-sharing. The paths of information flow between user and main memory, main memory and secondary memory, secondary memory and tertiary memory, are not only clumsy, but in many cases nonexistent. Furthermore, effective multi-programming of current machines is extremely difficult and causes much machine time waste. At present, good service to a few dozen simultaneous users constitutes the state-of-the-art, but the need is for good service to several hundred simultaneous users on a single computer system.

In the early days of computer design, the notion was to use a single program in a single machine to compute feverishly for large periods of time with almost no interaction with the outside world: today such a view is naive. The original notion has given way to that of a large multi-user, multi-processor, multi-channel system. Moreover, each user of the system asynchronously initiates jobs of arbitrary and indeterminate duration, which are subdivided into a sequence of processor and channel tasks. Out of this apparently chaotic, random environment emerges a public-utility view of a computation center. The multi-programming required for this time-sharing need only be performed once in the central supervisory program. Each user may thus profit by this efficiency without suffering a mitigation of the demand of his own particular program.

, 2

The tasks in this time-sharing system dynamically start and stop every few milliseconds. The memory requirements of these tasks similarly grow and shrink, and one of the major jobs of the supervisory program is the allocation and scheduling of computer resources. The general strategy of this supervisory program is clear. As each job proceeds, it is subdivided into tasks that are placed in a queue appropriate to a processor or a channel. The processors or channels are assigned new tasks as they either complete old tasks or are removed from them. All processors should be symmetric and capable of assignment to any of the computer tasks. A user should be logically and physically independent of the processors. Again, as with the processors, a user should be able to add or delete a channel according to system load or reliability without any reprogramming.

This system viewpoint offers a clear-cut approach to the multi-user, multi-processor computer. However, many subjects remain as interrelated problems and requirements: clocks, memory protection, program relocation, parallel tasks within a job, common simultaneous use of subprograms by many users, growth and shrinkage of program segments, and memory allocation. These considerations must play an important role in any determination of the hardware of a time-sharing system that can effectively satisfy large numbers of users with diverse problems.

<u>Research on the 7094 Time-Sharing System</u> - Robert M. Graham

The principal activity of the programming staff during the past six months has been to extend and enhance the Compatible Time-Sharing System being used on 7094's at both the Computation Center and Project MAC. Herein we shall summarize the major areas of change.

Twenty new public commands have been added to the supervisor. Of these, several are concerned with loading binary BSS subprogram files and manipulating library files. This allows a user to create and maintain his own private libraries with a high degree of facility and flexibility. A second group of commands was inserted to allow a certain amount of controlled group interaction within the system; several users working on the same problem can now manipulate files in a common file region. Finally, many of the new commands are programming language systems and bring the total number of languages available to ten. The Center's role in installing these languages was largely to encourage the maintenance of the programming systems. New languages and the people responsible for them are:

1. SNOBOL - Comp. Center
2. COMIT - V. Yngve
3. GPSS - M. Jones
4. AED - D. Ross
5. OPL - J. Weizenbaum
6. DYNAMO - J. Pugh
7. LISP - M. Minsky

A public file pool allows any user to place a file in it or to make a copy of any file in the pool. A user may thus pass useful programs on to other users.

In order to measure the performance of the system, a module has been added to the supervisor that collects status information. This module is executed every time the clock interrupts, currently every 200 milliseconds. All of the information defining the present state of the system, such as who is logged in, the status of each user, and who is currently executing, is kept in a common data block. The monitoring module, when it is called, transfers the current status information to a large, circular ring buffer. Any user may write a program to periodically empty the buffer and perform some analysis of the information in it.

A new login procedure has been implemented to provide greater security. After the user types the login command, the computer requests the user's secret password and turns off the printing mechanism on his console. The user then types his password, which is not printed, and the printing mechanism is turned on again while his password is checked by the supervisor program. After it has been verified, the supervisor checks the console identification to see whether the user is authorized to use that console. Finally, the supervisor prints the identification of the version of the supervisor being used, any message of current interest, such as a departure from the normal operating schedule, and a summary of the user's time and disk track allotments, all as the final step in the login procedure. An important administrative aspect of the new login command is that it includes the ability to have users in party line groups. Each group may have limits on the maximum number of members allowed access to the system at one time as well as on the acceptable stations to which the members have access.

Three new classes of remote consoles have been added to the system; the IBM 1050, the Telex, and the TWX prime. The Telex connection has been used to give successful demonstrations from England, Norway, Texas, and Washington, D.C. Both the Telex and the TWX prime are standard units and no special connection is necessary, so that it is possible for any unit on the international Telex or the national TWX prime networks to function as a remote console when the proper number is dialed.

The systems programming staff has been conducting weekly tutorial seminars. Individual staff members have given lectures and have led discussions about parts of the system they have worked on. The regular attendees are now quite knowledgeable on the detailed structure of the system. Notes are being written up to provide documentation. Moreover, proposals for additions and improvements to the system are being presented. Many valuable suggestions have been received from the attending users.

Several ways have been tried to enhance the documentation of the present time-sharing system. The "Time-Sharing System Notes" which were meant to supplement the basic CTSS Programming Manual, and the informal bulletins which notify users of up-to-the-minute changes, have been continued. To obtain some information from users about their opinions about the system, a REMARK command allows them to make remarks to the system; these are periodically read out of the computer and given to the Computation Center staff. The results of this command have been somewhat disappointing in that the remarks offered by users merely indicate a misunderstanding about how the system should behave. Nevertheless, this fact has been valuable, because it reveals weaknesses in the remainder of the documentation. In addition, system library subroutines have been documented largely by means of one page write-ups like the ones currently offered for the subroutines in the normal background system. Finally, an all-inclusive bibliography (CC-229) has been written.

Much time was spent on the evaluation of the potential of new computer systems for time-sharing applications. This evaluation assisted Project MAC in choosing the next hardware system. Meanwhile, the present IBM 7094 system still needs improvement. Particular features being worked on are: a new disk input/output control program suitable for multiprogramming, disk file maintenance and editing procedures to allow more convenient operation and better reliability, facilities for creating macro-commands, ability to initiate background programs from foreground, capacity for interconsole messages, and the ability to operate magnetic tapes from consoles in the foreground.

Research on Systems Programming Languages - Edward L. Glaser

This project's purpose is to perform research and advance development in software and in the interaction between hardware and software. We have investigated a basic systems programming language for Project MAC. This language must permit an efficient, powerful, but flexible programming of systems functions that is almost independent of the machine in which these functions are placed. Naturally, programs are differently coded on different machines; but, at an appropriate level, statements about systems policies are independent of specific machine structure and, consequently, are applicable to a large class of machine structures.

During the last six months, we have outlined the language. We also investigated various alternatives, before deciding to develop it. In form, it is an operational descriptive language that can describe not only other languages but also itself and extensions to itself. A property emphasized in the language is the facility to deal with highly complex but efficiently stored structures of data and programs. This type of language best abstracts those processes inherent in language definition and compilation and also those processes inherent in any large system supervision. It ought to reflect, in its abstractness, the processes we want in any computing equipment rather than a general description of all possible computers. Consequently, we have paid much attention to the description and handling of data as well as to the description and use of any highly complex operators that we might apply to these structures. The language has described a number of the specific languages used on Project MAC, such as MAD, FORTRAN, LISP, and COMIT; and, it can adequately describe a number of compiling processes used both here and elsewhere.

V

To aid this linguistic research, we have investigated translator compilers, particularly the CGS system table syntax compiler of the Air Force sponsored CL-II programming system. This compiler is a table-directed translator and is the most readily available flexible form of translation that we shall need during the development of the language. The compiler may not ultimately be the best way of translating the language, but it is an excellent tool with which we can extend our current programming powers.

In the next few months we shall have a first draft of the new language and shall use it to program the beginnings of a new time-sharing system; and, we expect to use this language to describe other high level languages more useful to individual projects. This work will determine the specific translation scheme to be mechanized for the next time-sharing system. In a subsidiary linguistic project we shall treat Braille as a formal language. While Braille is not so well formed as a programming language, as is also not so ill formed as a natural language; an interesting intermediate between the two, it will be an instructive model. This study is useful to Mr. Glaser, who requires a Braille translator to use the time-sharing equipment.

We are studying another way of constructing an executive program. An executive program is now a routine to which user programs transfer control either explicitly or implicitly through interrupts. A distributed executive program, that is, a set of service routines and of system tables and a local program which appends needed functions to user programs may be both feasible and useful. The CL-II system was developed by Computer Associates, Inc. to study such programs. The programmer should not be aware of the system's functions, and the system should behave as if a single executive program were handling system administrative functions of fault analysis, research analysis, resources allocation, and priority scheduling.

COMPUTER COMMUNICATION STRUCTURES

Computation Systems Development - Herbert M. Teager

The objectives of this group are primarily the design, the construction, and the evaluation of graphical input-output devices and languages. An ancillary interest is the development of models and of real-time techniques for information processing systems. The group is composed of Professors Teager and Stockham, graduate students Allan Scherr, Daniel Wilde, Edward Feustel, and W. R. Chiodo, and full time staff members Alexander Rutchka and Otis Wright, all of the Electrical Engineering Department. The activities of this group are described under the following headings: input-output devices, language development, supporting software, and system modelling.

I. Input-Output Devices

a. Graphical Input Table: We have acquired a prototype version from the Rand Corporation of a graphical input device conceived by this group. In its present form the device converts the location of a hand held stylus to a twenty bit position several thousand times per second. On the device we plan such modifications as an improved marking stylus, a transparent surface, a pseudo-keyboard, a fingertip input, and a blackboard sized input surface.

b. Graphical Input Interface: For the past year, we have been developing techniques that will make this device a primary input element for graphical, multidimensional language forms, such as circuit diagrams, line drawings, flow charts, and the ordinary multi-line representations of the formulae of mathematical physics. As a primary input element, this device requires both provisions for automatic recognition and buffering for telephone line transmission of a large (greater than two hundred) character and symbol font.

The form of the decoded characters will be a ten bit code and its binary location. The processing harware, already simulated, designed, and in the process of being assembled, is described in a forthcoming MAC memo.

c. Graphical Output: A group of specially buffered Cal-Comp plotters remains our primary media for graphical hard copy output. We have made its hardware compatible with the operation of the ESL Kludge and now use this equipment primarily to explore the minimum requirements for low performance graphical output systems. We are also gradually exploring the feasibility of low cost display and graphical output systems, derived from line-scan display and xerographic output systems that use UHF Television channels and multiplex among many on-line users.

II. Language Development

The primary real-time applications of current interest to this group are the analysis and simulation of linear and digital systems. For these applications new languages have been designed and reduced to software for active network analysis and digital system simulation; and, in addition, several existing ones such as BLODI (block diagram compiler) are being extended and modified at the input and output interfaces for graphical interaction. We are also developing for system analysis such tools as pole-zero manipulation and function transforms, such as the Fourier, Hilbert, and Convolution transforms. As the new graphical input-output devices and language forms become available, we shall systematically evaluate these building blocks of system analysis and shall explore the effects of machine reaction time, output plotting rates, and language compatibilities.

III. Supporting Software

In order to use graphical input-output devices within the CTSS system, one must write system programs to place user programs within reach and also specific subroutines to make these system programs more convenient to the user. In the case of the mechanical plotters and the Kludge, we are writing and modifying to suit the user a set of plotting and graphing programs. In the case of the graphical input device, we are writing a computer program to analyse the digest input character fonts for users who wish to use an on-line graphical input. We are also writing programs which will convert from the normal form of graphical input symbols, such as coded character, location, and extent, that occurs in random sequence into the common line-image form expected by most languages.

IV. System Modelling

A doctoral thesis in progress attempts to model the behavior of multi-processor computer systems in order to uncover and define controlling communication parameters; one is also in progress that attempts to abstract and analyze the operation of machine language symbolic programs. Finally, there is recently completed work that analyzes some invariant characteristics of handwritten symbols in order to provide design data for the on-line graphical input interface.

Research of Scheduling Algorithms - Richard Y. Kain and David J. Kuck

One of the important parts of a time-shared computer system is the scheduling algorithm that determines the order in which users are given access to the central processor. The general scheduling problem has two parts. First, a given computational job which a user presents at a console is best separated into smaller pieces either by the compiler or by the scheduler. Efficient techniques for such segmentation would prove valuable through a reduction of the overhead cost associated with swapping the program and data between the primary and secondary memories. Second, the smaller pieces into which the jobs have been split must be scheduled for the central processor to meet a chosen objective, such as the minimization of the mean waiting time which the user experiences sitting at the console.

We can handle the segmentation problem most effectively when we take into account some of the structure of the desired computation. Since the algorithms used to improve service must not consume more time than they save, segmentation is not feasible for a program written in assembly language. Some compiler languages are more suited than others for segmentation of the object program. We have been attempting to find compiler languages which, while easing the segmentation problem, are not inconvenient for the user.

One language uses "conditional" expressions, proposed by J. McCarthy, and "continued conditional" expressions, the latter our generalization of the former. The segmentation depends mainly on the flow of control through the program, rather than on some fixed quantum of computation time. This language is designed to emphasize the control flow aspects of the program and thus to simplify segmentation. To find a compact representation of the possible routes through the program, we must solve equations resembling those in regular algebra. When the program is nonrecursive, we can obtain the solution in closed form in the regular algebra. When the program appears to be recursive, a solution cannot be found in closed form, but conditions necessary and sufficient to guarantee the uniqueness of the solution of the equations can be found.

The second part of the scheduling problem is the scheduling of the segments for execution by the central processor. In the past, a systems programmer, by creating an algorithm which seems intuitively reasonable, has treated the scheduling problem in an ad hoc manner. Given a measure of value of the performance and some statistics of the user population, we may find a more systematic method of obtaining a class of scheduling algorithms. Hopefully, the optimum scheduling algorithm for given constraints would be relatively insensitive to small perturbations in either the value measure or the user statistics. Such an algorithm must also be inexpensively executable, that is, it must avoid combinatorial evaluations.

At present we are considering various value measures, such as the necessity of only small variations in the waiting time, so that the users could anticipate the quality of service. The smallest variations, for equivalent users, are associated with the round robin algorithm, which is known to be inefficient in swap time costs. We are also considering the possibility of on-line experiments to determine user reactions to various statistics of service.

Machine Structures - Jack B. Dennis

The machine structures group develops the design, analysis, and evaluation of advanced computer structures in accordance with multi-access computer systems. The research program of the group currently encompasses the following studies:

1. the development of models of program structure,
2. the characterization of interacting parallel processes,
3. the allocation and scheduling of computation resources,
4. the formulation and study of machine features and organizations,
5. the realization of pseudo-associative memories,
6. the design of terminals and their interaction with processes,
7. the development of a language for the practical description of information processing machines and on-line

programs for their manipulation.
The paragraphs that follow contain summaries of the accomplishments and plans in the studies listed above.

Models of Program Structure: We pointed out in a recent Project MAC proposal that a model of program structure is necessary for the evolution of new computing structures and for the development of adequate principles for the allocation of computing resources. The group has spent considerable effort on this model. In multi-access computer systems, we have extended and clarified the concepts of segment and phase, introduced by Holt, and have introduced the concepts of process and of sphere of protection. We have also recognized and examined their properties as an allocatable computation resource. Projected work of the group will examine the hierarchical structure of executive functions and will elucidate relationships, such as the nature of intercommunication, among spheres of protection.

Parallel Processes: The success of executive function in a multi-access computer depends upon the successful organization of concurrently operating processes; and, procedures of interest to users frequently involve several concurrent processes. At present the properties of interacting parallel processes are poorly understood. A first study of this problem together with an attempt to formalize some of the notions occurs in an unpublished paper by Hans Witzenhausen. We shall continue his work and relate it to the segment structure of programs.

Allocation and Scheduling: We have developed a view of allocation and scheduling in a MAC system that splits these functions into two parts, policy formulation and policy execution. The policy formulation component monitors the policy execution component and modifies its parameters so as to achieve a fair distribution of computation resources. We have examined the scheduling algorithm developed for CTSS and found it to allocate and schedule in a similar fashion. We envision much of the policy execution component implemented in hardware for future MAC systems as the system response becomes faster and faster. These concepts enable us to evaluate machine features as aids to the allocation and scheduling function and to formulate an economic theory of computation resources.

Machine Features and Organization: We are studying those multi-processor computer configurations particularly appropriate for MAC systems. We have already studied a simple class of multi-processor, multi-memory systems both by simulation and analysis of the properties of queues that arise at memories.

A paper design for the order code and register structure of a processing unit, known as MAP-1, is nearly complete. The design has a syllabically structured order code that both permits pure procedure coding and implements the addressing, by name, of procedure and data segments. We have also devised and improved a technique of implementing, by segment name, memory references that arise from the execution of a process. The present version has two levels of associative search. One determines the legality and type of a reference to a particular segment by a process that operates within a certain sphere of protection; the other translates a segment name and page number within that segment into a physical block address at the page location in the main memory. This technique also makes possible flexible allocation of main memory. We shall also study the automatic assignment of processing units to processes, system configurations without full interconnection of memories and processors, and reliability.

Associative memories are desirable components for certain functions in large computer systems, such as for the implementation of memory references by segment name. True content addressable memories have undesirable search times and cost properties. We may implement associative searches by using conventional coordinate addressed memory together with a hash addressing technique. In many applications such a pseudo-associative memory performs better than truly associative hardware. We have devised several schemes of implementation in connection with the storage mapping function. We shall continue work towards an understanding of the properties of key transformation and of the limitations of the general technique.

Grapheme-to-Phoneme Translation of English - Francis F. Lee

The pronunciation of English words by an English speaking person depends partly on rote memory and partly on rules which operate on substrings of each word. To state these rules with absolute clarity has not been possible. The existence of rules anticipated by intuition is demonstrated by the observation that the pronunciation of a list of English-like nonsense words by a number of test subjects does not vary significantly and that certain morphophonological rules begin to develop in the minds of children between the ages of four and seven.

The objective of our research is a study of the structure of rules relating English orthography and pronunciation. An application of this study is in the design of a reading machine for the blind and of a computer-to-man speech feedback and also, perhaps, in the elimination of the dichotomy of the look-and-see method and the phonic method of teaching English in the primary schools. To achieve this study, the investigator first analyzes the relation between the spelling and pronunciation of a limited but most frequently used subset of all English words, then synthesizes rules based on this analysis, and finally tests these rules on an additional subset of English words.

The work accomplished in the period from February to June 1964 has been primarily test preparation. The staff of the computation center has placed into time-sharing the language, SNOBOL, developed by the Bell Telephone Laboratories. The investigator has performed extensive experiments on this language and has found it suitable for this study; and, he anticipates those further changes in the language that will make it a more powerful tool. The eighteen thousand most frequently used English words are now entered for machine processing. Each entry contains the word, the relative frequency of occurence, and the various accepted standard American pronunciations. The processing of these words used SNOBOL to correct errors in transcription and inconsistencies in the stress markings. Each monosyllabic word, about one in every six in the data, is now separated to facilitate future initial and final consonant cluster studies; and, a tentative algorithm classifies graphemes (y) and (w) as either glides or vowels or parts of dipthongs.

The experimental phase will continue with the formation of the consonant and vowel clusters for both the grapheme words and their phonemic counterparts. Since many words have multiple phonemic entries, these entries must preserve as much information as possible; at the same time, they must not overburden the data processing equipment. A study of the initial and final consonant clusters of monosyllabic words shall determine their applicability as clues to detect morpheme boundaries. Plans for the synthesis and testing part of a translator program that will convert the grapheme-to-phoneme translation rules into SNOBOL language form the subject of a future report.

Simulation of Large Computer Systems - Allan L. Scherr

The objectives of this research are the development of a simulation technique for the study of large computer systems, the validation of this technique, and the generalization of its results. In particular, the aim of the simulation study is a class of real-time time-shared computer systems similar to those in operation, such as CTSS. The principal objects of study are:

1. suitable models which will yield insight into the operation of the above systems, results of the proper form and accuracy to achieve this insight, and economy in running the simulations,
2. the validation of these models through a comparison of their results to the experimental data obtained from CTSS,
3. the generalization and simplification of the models and results through, among other ways, a reduction of the dimensionality of the variable space. A simulation programming system is being developed as a necessary tool for this research.

During the past six months the simulation programming system has been so written and debugged that it is a useful tool. We have simulated initial models and are studying the results in order to simplify and generalize the models. Now that experimental results from CTSS are becoming available, we shall use them to verify the models.

Simulation of Multiple-Access Computers - Earl C. Van Horn

During the first half of 1964, we conducted research on four topics: the simulation of multi-processor, multi-memory computer systems, the analysis of multi-processor, multi-memory computer systems, the structure of multi-user computer systems, and the design of the structure and order code of a processor. We have studied a simple type of multi-processor, multi-memory system both by simulation and analysis. On a given memory cycle each processor that is serviced on the previous cycle independently selects a new memory from a uniform distribution of all the memories. The treatment of queues of processors at a memory is first-come, first-served; and, processors remain in the queue at their selected memory until they are serviced.

The results of our simulation studies are frequency histograms of two variables, the number of memories active during a cycle and the number of cycles required for a processor to make an access. The following results, typical of those obtained from the simulation program, are the average over five runs of 5000 cycles each of a twenty-processor, twenty-memory system. The histogram of the number of active memories is a bell-shaped curve having a mean of 12±.04 and a standard deviation of 1.48±.02. The histogram of the number of cycles for an access is a monotonically decreasing curve with a mean of 1.66±.01 and a standard deviation of .96±.02. The tolerances associated with these results indicate the total variation of the quantities over the five runs.

An interesting result of the simulations occurs in the cycles per access histogram. In almost every case, the most probable event is one cycle per access, even in a system of eighty processors and twenty memories. Reflection on this result has led us to suspect that long queues tend to build up at certain memories and that service is very slow at these memories and very fast at others. In further research we shall try to measure this suspected behavior and determine whether certain dependent memory selection distributions can reduce the system's tendency to form long queues.

We have analyzed the simulated system as a Markov process with, as state variables, (p+1)-tuples of the following type,

$$(n_0, n_1, \ldots\ldots, n_p),$$

where n_k is the number of queues of length k in the system. We have found a recurrence relation that gives the number of states necessary to describe a system of p processors and m memories. In addition, for the determination of transition probabilities among the states, we found a method which takes advantage of the fact that each of the above states corresponds to a group of states in an auxiliary Markov process. The state variables of the auxiliary process are m-tuples of the following type,

$$(q_1, q_2, \ldots\ldots, q_m),$$

where q_k is the length of the queue at memory k.

If we treat these systems as Markov processes, we can state with certainty the type of behavior expected in them under a variety of conditions. We shall use Markov analysis to characterize the transient behavior of these systems. Since the Markov analysis of a twenty-processor, twenty-memory system requires 623 states of the first type, Markov analysis cannot completely replace simulation as a method for studying multi-processor, multi-memory systems. Moreover, the methods used in the analysis thus far are not readily applicable to systems which have dependent memory selection distributions and more practical queue disciplines.

We have also carried out research on the structure of multi-user computing systems. The sphere of protection is an important system entity, which we may define roughly as the set of all system resources to which the user may make reference. Sphere creation and deletion activities produce a natural hierarchy of spheres. We are attempting to refine the concept of an executive program, to discover the natural relations among multiple executives, and to find a correspondence between these relations and the sphere hierarchy.

We have already developed the hardware enforced lockout of a memory segment to ensure the exclusive use of one processor, and we are now investigating intersphere communications and processor allocation. Design effort is substantially complete on the nonexecutive order code and register structure of a processor called MAP-1. This processor features a syllabically structured order code, programmed addressing, and ten variable size general purpose registers. We have specified a rudimentary assembly language for writing MAP-1 programs. The MAP-1 order code, which has an algorithm encoding efficiency that compares favorably with that of the 7094 and 360 order codes, is designed to facilitate the coding of pure procedures within a segmented data structure.

Measurement of Multidimensional Transducers - Jay R. Sklar

The Fano sequential decoding algorithm has been applied almost exclusively to the decoding problem for tree encoded messages. Its success in that area, however, does not hinge in any way on the communications aspects of the problem, but rather depends only on the tree like nature of the encoding process and on certain properties possessed by the metric that we must define to carry out the algorithm. The generality of these conditions makes desirable the determination of those noncommunications tree search problems susceptible to search by a similar method. We have written, debugged, and operated in CTSS a program that performs this search.

Two conveniences have evolved from the intimate communication between man and machine possible under on-line operation. First, the tree search is not chiefly a series of numerical computations but, instead, a large number of logical operations based on a few simple numerical calculations. It is, therefore, a dynamical system whose operation is best understood through step by step contact with the logic. We may dynamically represent this system by using the ESL display console as a direct output of the simulation. Second, of the many interactions among the various parameters of the tree search, some are quite subtle; and, by analyzing the results of the simulation, we can detect those subtle ones and then investigate their details. At present we are using the program to obtain information on the relationships among the search parameters in various tree search problems.

<u>TREET: A New List-Processing Language</u> - Edward C. Haines

My objective is the design and implementation of a new list processing language to supersede existing languages. The advantages of this language, which I call TREET, over existing languages will be ease of learning and use, efficiency in algorithm expression and program compilation, and potential efficiency in operation. I have defined most of the basic aspects of TREET. It is essentially similar to LISP, but differs from it in nomenclature, formats for writing program statements, treatment of error conditions, comments, a more powerful GO statement, and the definition of several additional functions. LEAF is a function of two arguments: the first is a symbol, the second a tree. If the symbol is a leaf of the tree then LEAF returns to and has as its value the corresponding node: if otherwise, it has NIL as its value.

The language needs slight modifications to conform with the input requirements of LISP; it differs from it principally in punctuation. A new LISP pseudo-function compiles the program into a format similar to the Program Feature S-expressions. The standard LISP interpreter and a new interpreter function then interprets this format. Since this mode of procedure is completely compatible with the present LISP system, a user may freely intermix functions defined by all methods. However, the TREET language is not dependent upon LISP, and with suitable compilers or interpreters it can operate in a somewhat different environment. For the present implementation I find LISP convenient. I am now attempting to define various of my notions with respect to programming in LISP and, thus, to confirm them.

S-PLANE: A Program for the Manipulation of Zero-Pole Patterns

- Edward Feustel

S-PLANE is an experiment in data storage in which the structure of the storage permits efficiency of operation. It is an educational and industrial aid that displays and analyzes certain functions of a complex variable. An experimental list structure of specialized design like that used in "Sketchpad, A Man-Machine Graphical Communications System" by Ivan Sutherland of Lincoln Laboratory, was developed to simplify the presentation of rational functions in storage. Various control characters on the face of an oscilloscope allow appropriate subroutines to alter storage and data values, to initiate calculation of frequency responses, and to combine functions. The log vs. magnitude plots of amplitude which are displayed give the operator insight into the process of synthesis and analysis of linear networks. Displays are provided with dynamic calibration to permit accurate experiments.

Although S-PLANE is an experiment capable of being improved by reprogramming, it has engendered considerable interest in its early use. The program now consists of three sections: subroutines for the generation of the specialized storage, subroutines for arithmetic action, and subroutines for display and program control.

104

ARTIFICIAL INTELLIGENCE

✓

<u>Derivator</u> - Marvin L. Minsky

Derivator is a PDP-1 program that permits examination of the solutions to differential equations of the form:

$$dy/dx = f(x,y)$$

by inspection of a visual display of the trajectories. Because it employs fixed point arithmetic to maintain visual display speeds, Derivator must be regarded as a qualitative tool. It is subject to truncation error in the trajectory following program and round off error due to underflow in the function definition programs for <u>dy</u> and <u>dx</u>. Still, it seems suitable for the study of the topology of the solutions around singularities, etc. The input to Derivator must be written in PDP-1 machine language with DDT. For many cases of interest, however, that writing is easy.

The controls of Derivator are sense switches, the light pen, and program parameters that may be typed in. At the start of the program, the scope face appears as in Figure 4. Somewhere in the field there is a small bright circle; this surrounds the initial condition point (qx,qy). We may move this freely with the light pen. Several options are available:

1. the display of the entire direction field,
2. the magnification of a selected portion of the field,
3. the application of a rotation matrix to the tangent vectors. (If a rotation of 90 degrees is selected, then equipotentials and gradients are interchanged.)

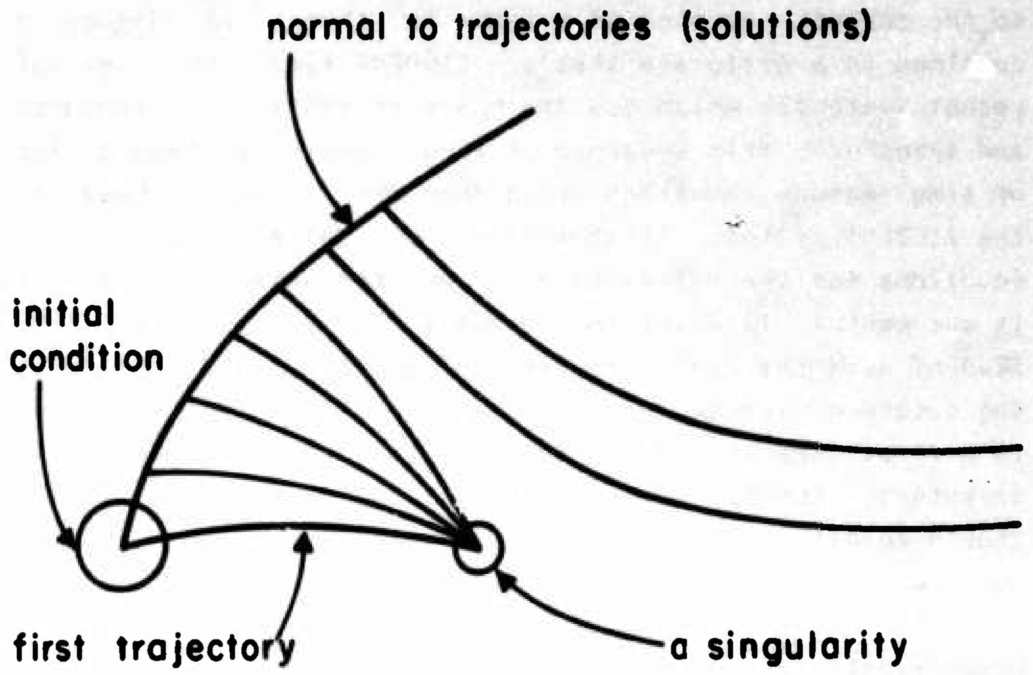


Figure 4. Initial Derivator Display

Computer Problem Solving with Natural Language Input - Daniel G. Bobrow

The STUDENT problem solving system, programmed in LISP, accepts as input a comfortable but restricted subset of English which can express a wide variety of algebra word problems. STUDENT finds the solution to a large class of these problems. It can utilize a store of global information not specific to any one problem and make assumptions about the interpretation of ambiguities in the working of the problem being solved. If it uses such information or makes any assumptions, STUDENT says so to the user.

The linguistic analysis in STUDENT is a first approximation to the analytic portion of a semantic theory of discourse outlined in a doctorate thesis. STUDENT finds the set of kernel sentences whicn are the basis of the input discourse and transforms this sequence of kernel sentences into a set of simultaneous equations which form the semantic base of the STUDENT system. It then tries to solve this set of equations for the values of the requested unknowns. If it is successful, it gives the answers in English: if not, STUDENT asks the user for more information, and indicates the nature of the desired information. The STUDENT system is a first step toward natural language communication with computers. Further work on the proposed semantic theory should result in much more sophisticated systems.

Mechanization of Proofs in Number Theory - David C. Luckham

The problem with existing theorem proving programs for first order logic is that the time required to prove an interesting theorem is far too great to be practicable. These programs are based on inefficient logical proof procedures that generally take irrelevant steps in their search for a proof of a given theorem. Furthermore, the time they take to decide whether there is a proof usually increases exponentially with the number of steps in the supposed proof. We have made improvements in the efficiency of programs based on Herbrand type proof procedures by using a combination of logical techniques that reduce the number of propositional clauses considered. We are interested in finding techinques to improve the power of existing proof procedures when the area in which proofs are sought is restricted to a particular mathematical theory that is formalizable within first order logic, in our case, to Elementary Number Theory.

In building a theorem proving system for Number Theory we have two methods of approach: we may set out to derive very elementary properties, such as the commutativity of addition from the basic axioms, or else we may start the system with an organized knowledge of such elementary properties to obtain a basis for derivations of less trivial theorems.

The second approach appears to us the more fruitful to add to our experience of techniques for the formalization of definitions of arithmetical functions and predicates, for the handling of special arguments, such as induction and counting arguments, that recur in proofs in Number Theory, and for the selection chains of intermediate lemmas that lead to a complete proof of a given theorem. It is this experience that will eventually lead to the construction of a good stock in trade system for Number Theory.

For this construction, we have analyzed the formal proofs of a number of theorems from the Theory of Congruences in order to build up the backlog of elementary functions for derivation in this area. Here are some examples:

1. If $a=b \pmod m$, then $ac=bc \pmod m$.
2. $a=b \pmod m$ and $c=d \pmod m$ implies $ac=bd \pmod m$.
3. If $ar=br \pmod m$, and d is the g.c.d. of m and r, then $a=b \pmod{m/d}$.
4. Fermat's little theorem: If p is a prime, and p does not divide a; then, $a^{p-1}=1 \pmod p$.
5. Euler's theorem: If (am)=1 then $a^{\phi(m)}=1 \pmod m$ where ϕ is is Euler's Phi-function.

We have thus far programmed in LISP several parts of the theorem proving system, but we have not yet debugged or run them. The first of the two proof procedures for first order logic that have been programmed is that of Davis, McIlroy, et al., which is a Herbrand type procedure that incorporates both a necessary condition for a conjunction of propositional clauses to be false, the linked conjuncts condition, and for a choice of the next proof step in certain heuristic methods. We have modified the procedure by adding Complementary Literal Elimination. The second procedure is the Resolution procedure of J. A. Robinson.

We have also programmed axioms for number theory, definitions of a number of Skolem functions, such as lx-yl, exp (x,y,), and g.c.d.(x,y,), and predicates, such as prime (x). These definitions are made in free variable form. The list of axioms contains much more than Peano's postulates; it contains, for example, a form of finite induction with which we may conveniently handle the counting arguments in proofs of theorems, such as 4. and 5. above. In the future we may add further theorems and function definitions. We have also been working on programs for equality and for the associative, commutative, and distributive laws. Our object is to save the time the proof procedure spends in deducing the consequences of the axioms of equality or of the elementary properties of numbers.

We shall compare the performance of the two proof procedures on various classes of theorems in first order logic. Robinson's procedure seems to differ from the usual Herbrand type procedures. We cannot adequately compare these procedures, but we suspect them to be of roughly the same power, although the procedures of Davis, McIlroy, et al. may prove superior on theorems whose statements contain a large number of clauses. One of these two procedures will form the basic proof program for the system. Second, to obtain the derivations of Number Theory theorems, we shall give the system a chain of lemmas that lead to a proof. We allow previously proven lemmas as axioms in subsequent proofs; and, thus, we shall be able to obtain proofs of easy congruence theorems, such as 1. and 3. above. Derivations of 4. and 5. may not be easy for us to come by, even with use of a suggested chain of lemmas; hence, they should afford a good comparison of methods for the handling of induction and counting arguments. If and when we manage to obtain proofs, we shall experiment with methods whereby we may choose and may possibly obtain nonstandard proofs by varying the backlog of axioms and functions.

The Mathematical Assistant - William A. Martin

The Mathematical Assistant will give on-line assistance to mathematical analysts. Our initial effort is in the approximate solution of nonlinear differential equations by means of asymptotic expansions. We have written routines in the LISP programming language to carry out tedious transformations and have completed programs for simplification, differentiation, collection of terms, expansion in a Laurent series, multiplying out, substitution of a single equation for an unknown, and polynomial manipulation. To accomodate these routines in the time-sharing--nvironment, we have extensively modified LISP by adding routines for communicating with the teletype and the disk along with extensive facilities for editing and debugging. With chaining, we can communicate numerical work to MAD programs and so make available to the Assistant the SHARE library.

Since the user is to direct the LISP routines through the PDP-1, we have written a program to display mathematical expressions on the PDP-1 scope. In the course of writing the LISP routines, we developed a system for the hash-coding of functions of a complex variable by means of finite field arithmetic. This system allows us to perform the frequent operation of expression matching by probabilistic means rather than by canonical ordering.

In the future we shall expand the LISP routines to handle transformations on a vector space and to write MAD programs. We shall develop routines for the classification of expressions in conjunction with a language for the specification of tree search algorithms and also executive functions necessary for the operation of the system through the PDP-1.

Integral Table Look-Up Procedures - Joel Moses

In the course of extending the area studied by Slagle on heuristic symbolic integration, I have constructed and studied two programs for integral table look-up procedures, both of which use the heuristic of evaluating the integrand at a predetermined point and of searching in the table for the expression having the same value. The first look-up procedure can handle only a restricted class of expressions. A basic restriction in it is that arguments of trigonometric functions must be simply the variable and not a function of it. This look-up procedure can recognize a large class of algebraic equivalences. In particular, it recognizes that

$$1 - 1/1+\tan(x)$$

is equivalent to sin(x), and thus its integral is -cos(x). This ability is not without cost since the procedure may see equivalences where none exist. There are several means of making the error rate smaller than the present one of 10^{-8}, but we have not made use of them.

The second look-up procedure can handle a larger class of input expressions and will not yield an incorrect integral; but, it will not check for many equivalences. This procedure is suitable for searching a standard integral table and uses an evaluation routine which assigns to those literals and constants much used in the intergral table the same value in similar situations. Thus sin(2x+3) is made equivalent to sin(4nx+3y+3.2) and to sin(cx+d), but not to sin(2x) and not to sin(x+3). If a table expression should have the same value as the input, then the expression and its integral are read from the disk. I use a routine borrowed from J. Slagle's program to match the input expression and the table expression and to obtain a list of literal values which will make the two expressions equal. When I substitute these literal values in the integral, I produce the result.

A Geometry Theorem-Proving Program - Timothy P. Hart

"The demand is not to be denied; every jump must be barred from our deductions. That it is hard to satisfy must be set down to the tediousness of proceeding step by step. Every proof which is even a little complicated threatens to become inordinately long." G. Frege, 1884

During the last half of the nineteenth century the need for formal methods of proof became evident to mathematicians. A desire for rigor has persisted since that time and stimulated knowledge of formal methods; but, for the tediousness noted by Frege, little mathematics uses fully these formal methods. We hope to use computers to take the drudgery out of formal demonstrations, just as we now use them to take it out of accounting. An intriguing prospect, however, is that computers will eventually both devise and prove nontrivial theorems wholly on their own. While we have not even attempted the invention of interesting conjectures, i believe that we shall soon achieve the mechanical proof of nontrivial theorems.

A.I. Memo 56, "A Proposal for a Geometry Theorem Proving Program", contains a state of the art summary of the machine theorem area. Since the writing of this paper, J. A. Robinson has made an important advance which is described in a forthcoming paper in the JACM. I have incorporated this work in a running program that is the best machine theorem prover I know. In its current state it can not quite prove the second version of the theorem in my proposal: an isoceles triangle has two equal angles. In this second version the machine must first decide to drop a line from the apex to the midpoint of the opposite side and then show that the two smaller triangles so formed are congruent. It formulates the latter demonstration, but runs out of storage before achieving it. The present program is quite crude. I expect to be able to improve its performance, for instance, by having it use the diagram heuristic in it. The goal of my project is machine produced proofs of the theorems in Forder's "The Foundations of Euclidean Geometry", which is a formalization of geometry up to the high school level that starts from a small axiomatic basis.

File Maintenance and Syntax Generalization - Ian C. Pyle

The FILEDT program for file maintenance is a simple conversational program which provides useful facilities for the listing, renaming, or deleting of files from the users' file directory. The conversational style of programming makes sense only in a time-sharing experiment. It codes the program to make requests of the user for input, which he need not present in a predetermined order with the consequence that, for example, whenever the program discovers an error in the input, it points out the error and asks for the input again. The program also contains its own operating instructions, which the user may have printed out.

The object of syntax generalization is the creation of a program which, when provided with examples of the constituents of a language, will determine the syntax of this language. The problem, as stated above, is not well-defined because clearly the finite number of finite strings by means of which we specify the constituents of the language cannot completely define it. The syntax might be only those strings given as examples, or it might even be all the strings of bank symbols. A subsidiary problem is, therefore, the formulation of criteria which we may use to choose from among various syntaxes consistent with the given examples. We can most effectively limit the generated syntax by allowing the data of the program to contain, together with examples of allowed constituents of the language some counterexamples, that is, strings which are not constituents of the language.

An object language is phrase structured when it contains classes of strings, or phrases, which we may substitute for one another without affecting the correctness of any string containing them. Each phrase of a phrase structure object language is then a sublanguage. When the data specify the currently exemplified phrase, the examples given for each phrase represent a primitive syntax for the phrase. A simple analysis will determine where an example of one phrase is embedded in another. Since a maximum for the number of forms is set by the number of examples given, a criterion for a desirable syntax is that it combine as many as possible of these phrases that are compatible with known counterexamples.

Theorem Proving on a Time-Shared Computer - L. Seligman

Our current research investigates the use of a time-sharing computer as an aid in theorem proving. We have written a program in the LISP programming language for the MAC 7094 which permits the user to type in an informal proof and then proceeds to type back a mechanically verified formal proof; and, we have created a special macro-language with expressive capability beyond that needed in this application. Since the logic system in use is quite powerful and especially convenient for use with a computer, extremely difficult proofs will be within the capabilities of the system.

Previous work in theorem proving on the computer falls into three general categories. The largest of these, research into purely mechanical proof generation, uses a specially coded proof procedure algorithm and will produce eventually a proof where one already exists; but, usually, it will search indefinitely where one does not exist. These procedures require both storage space and execution time that grow faster than the exponential with increasing proof complexity; and, thus, they have been unable to prove difficult theorems. The search for better algorithms led to the basis of the proof system used here. This system is especially suited to this application since it couples a reduction in the number of necessary steps with an increased amount of purely mechanical work.

Heuristic theorem provers have done little theorem proving, but they have yielded insight into the basic processes of theorem proving. The major handicap to their further development is the difficulty in programming new heuristics, since no general framework for these heuristics exists. We hope this research will aid in the development of that framework. It will certainly provide a set of conventions and a library of the basic procedures necessary.

The third category, that of proof checking, is most directly related to current research but has few results yet reported. Inconvenient proof systems have led to requirements for machines far larger than any currently available; and, thus, further research in this area is presently limited. The advent of time-sharing has made possible a change in the nature of proof checking.

The notion that a computer can work together with a man in proving theorems is central to this research. The proof system to be used permits a man, with the specification of but a few steps, to guide the computer to a proof, the tedious work of which is done by the machine. The macro-language is the key to this process, as it provides a method of writing simply procedures which permit the execution of proof steps, such as "repeatedly do set x of simplifications". One example of a macro-language proves a number of cancellation laws of number theory, while another is essentially a decision procedure for symbolic logic.

We are proving currently a large number of theorems of number theory using this program. The theorems we have already proven are far more difficult than those previously mechanically proven or verified. Some we may yet be able to prove are more elaborate than those proven in any formal system. We are concurrently refining the program and cleaning up its printout to make it suitable for general use by others. The future extension of the program into metamathematics is our distant goal.

A Language for Binary Relations - D. P. Bovet

This work describes geometric properties of figures with a set of binary topological relations. Such a description, though incomplete, is useful in pattern recognition problems. Binary relation Syntax (BIRESY) is a LISP program which deals with sets of binary relations. It has two parts:

A. a predicate that decides whether or not a set of relations belongs to some list of topologically incompatible couples of relations,

B. some algorithms that, given a set of relations that represent a figure, obtain a new set which is a canonical description of it.

Part A. of BIRESY is a theorem-proving program; with a given set of relations, it generates a list of induced relations and tests every newly generated relation with respect to a list of incompatible relations. A set is said to be compatible if none of its relations or generated induced relations is incompatible with the remaining ones. A finite number of relations always permits a determination of compatibility. Part B. of BIRESY is a set of routines each of which transforms a set of binary relations into a new set with some canonical properties. Examples of sets with canonical properties are descriptions with a minimal ordered number of relations and descriptions with figures ordered by the number of relations they share with others.

<u>M-Expression Translator</u> - Gloria Bloom

This program, written in METEOR, accepts statements written as M-expressions and produces as output the corresponding S-expressions; the terms M-expression and S-expression are defined in the LISP 1.5 manual. The principal advantage to the programmer in using M-expressions rather than S-expressions is that they seem to him more like conventional mathematical notation and are therefore easier for him to learn to use and to understand when he examines them at a later date. Counting parentheses, a chore which can take a significant amount of time in the writing of a LISP program, need no long burden the programmer.

3 [illegible]

For example, for the S-expression,

```
      (COND  ((GREATER P (PLUS A (TIMES B C)) 0) T)),
```

there is the equivalent M-expression,

```
                    (if A+BC>0  then T).
```

The programmer may define arbitrary equivalences between symbols and also define any symbol as an operator. He can change the precedence level of any operator and thereby affect the order of processing of that operator. This program requires an I/O routine for the interpretation of the twelve bit character mode and also a golf ball with a richer character set than the present one.

LIBRARY RESEARCH

Technical Information Project

Search Procedures and Measures of Relatedness

12

<u>Technical Information Project</u> - Myer M. Kessler

The Technical Information Project at M.I.T. uses modern computer and communication technology to facilitate access to scientific and technical literature. We are establishing a pilot information system for real-time interaction between a wide variety of users and their literature. For every article we key punch the title, author, institutional connection, and bibliographic notes given by the author. We may include other indexing information from time to time.

We have written a set of programs for flexible and easy communication between man and computer in a search of these articles. The programs are:

1. BROWSE: Starting with any given page, the computer will browse serially through a given volume for whatever information we request.
2. SCAN: In a given journal-volume region this program scans the title, author, location, or bibliography for a given item. If it locates the item it can print, store, or count whatever we request of it. For example, it can find all articles with the word "cryogenics" in their titles.
3. SHARE: This program searches a broad segment of literature for articles that share some property with a given article; for example, it can find all the articles that share one or more citations with a given article.
4. MATCH: Starting with an article of known use and function, MATCH examines the literature and finds the articles that constitute the best match to it.

In addition to writing the above programs directly concerned with literature search, we are also working on a number of statistical studies of historical and general interest.

We have recently developed the file structure and program facilities to an extent that they permit some limited engagement with a user group, and we are using these developments to improve the coupling between man and the computer. The computer library and the programs will, we hope, be available to some MAC users early in September, 1964. This library will be the first sizable one in a form usable with a computer. Monitored results of this development should increase our understanding of this computer function and guide further system development.

Easy access to the literature is a significant research tool. It frees scientific creativity as do other experimental and theoretical research facilities. The success of this computer application as an aid to scientific thinking will depend on a careful matching of the habits and speeds of intellectual processes to machine processing and displays. Our present facilities are of a size and scope as to fall within both real and real-time situations. The next part of the project is the development of a language and display capability for broad and flexible communication between man and computer, perhaps using both audio and visual displays.

Search Procedures and Measures of Relatedness - Evan L. Ivie

Current library methods associate documents together in the same subject category. Mechanized retrieval systems make possible much more precise measures of the relatedness of documents. R. M. Fano has suggested one such measure derived from information theory that can associate documents. The objective of this program is the development and evaluation of measures of relatedness between documents and the design of suitable procedures by means of these measures.

A test file of sufficient size and interest to attract serious users is necessary. The Technical Information Project is already accumulating such a file under the direction of M. M. Kessler. It currently consists of about 35,000 articles from fifteen physics journals and is growing at the rate of about 1000 articles and one journal a month. The information available on each article is the journal identification code, volume and page numbers, title, author(s), author location(s), bibliographic citations, and subject index categories.

To expose the data file to an environment of users with real problems, we are using CTSS and the other facilities of Project MAC. We shall restrict the initial test system to the use of a standard MAC console, and possibly a graphical display device. The basic programming language is MAD; but, the list processing techniques of languages such as SLIP and COMIT will be liberally used.

We may define information retrieval as the interaction of a user with a document collection. Let us assume that this interaction results eventually in a partitioning of the document collection into two disjointed subsets: one containing those documents of interest ; the other, those not of interest. Fano and Watanabi defined an information theoretic correlation coeficient which is a measure of the degree to which a set of events $x_1^1, \ldots, x_r^1$ are correlated:

$$C(x_1^1, \ldots, x_r^1) = \log \frac{P(x_1, \ldots, x_r^1)}{P(x_1^1) \ldots P(x_r^1)}$$

Fano suggests that if x_i^1 represents the event that the document is relevant, then we may regard the correlation between the events $x_1^1, \ldots, x_r^1$ as a measure of the mutual pertinence or relevance which exists between the documents 1 through r.

In order to calculate C for any arbitrary set of documents selected from a collection of n documents, we should have to estimate and store at least 2^n-1 probabilities, a number out of question for any document file of reasonable size. If we are to use C, we must simplify.

We may, by ignoring higher order terms, approximate the correlation between any two subsets of events:

$$C[(x_1^1 \ldots x_r^1)(y_1^1 \ldots y_r^1)] \approx \sum_{i,j} I(x_i^1; y_j^1)$$

We must estimate and store, at most, n univariate and $\binom{n}{2}$ bivariate probabilities. By approximating the correlation coefficients, we may describe our problem in terms of a network of document nodes with positive and negative correlation links connecting them. A formal theory can cover such a network. The beginnings of such a theory we have reported elsewhere.

Concurrently with the development of a formal theory that covers the correlation network, we are experimenting with various methods for finding document clusters. We hope by these means to find an efficient procedure that will produce the appropriate cluster for any given input request, and we are now studying and elevating several clustering methods. The retrieval system would be helpful if, in addition to supplying answer sets in response to valid requests, it also detected improperly specified requests. We have analyzed two types of improper requests and have developed methods for detecting them.

The final problem discussed here is the estimation of univariate and bivariate probabilities from which the correlation coefficient is derived. Conceptually, the best way for us to determine these probabilities would be to have available a large population of users and be able to monitor their interaction with the file. We could then make the following counts and estimates:

N: the total number of usages of the collection,
N_i: the number of usages of document i,
N_{ij} : the number of joint usages of documents i and j.

However, even with such a user population available, a number of problems would still exist. For example, much use of a document twenty years ago would automatically give a high probability of relevance to that document today. To counteract this high probability of relevance, we may apply a suitable decay factor to the probabilities of the system to account for the relatively higher interest in current literature. Also a newly published document has no usage history and will hence have a zero probability of being relevant. We must supply some way of connecting such a document into the system so that it will develop a usage history. All of the documents in the initial collection have the same problem. To solve this problem, we are examining various types of pseudo-users to see how closely they simulate real file usage.

ELECTRONIC SYSTEMS LABORATORY

Display Systems Research

Computer-Aided Design Project

Input/Output Equipment for the PDP-1

12

Display Systems Research - John E. Ward

A. Introduction

As part of the work on the Computer-Aided Design Project, which is being conducted for the Air Force Materials Laboratory under Contract AF-33(657)-10954, the Electronic Systems Laboratory has for several years been investigating the design of a man-machine interface which was suitable for use with commercial computers and which would permit that type of on-line manual input and manipulation of graphical data necessary for a computer-aided design system. Existing display systems, such as the TX-2 scope used for Sketchpad, had a desirable graphical interaction, but the data handling for the display tied up a substantial fraction of the capacity of the associated computer. Therefore, we sought to transfer the major part of this computational load to specialized computing circuits in the display system and at the same time, by placing all display operations under program control in the general-purpose computer, to maintain the flexibility of our system. Early in 1963 we had completed design studies, including simulation on the EE Department PDP-1 Computer. An initial version of the equipment was placed in operation on the MAC 7094 in a non-time-sharing mode in November, 1963. In January, 1964 an Input/Output Adapter, part of the A-Core Supervisor Program, was written; and, it has permitted subsequent operation to be within the time sharing system.

The ESL Console shown in Figure 5 is installed on the ninth floor of Technology Square adjacent to the Project MAC 7094 computer. We purchased the basic display tube and its associated deflection circuitry as a specially engineered version of a standard commercial display unit, the Digital Equipment Corporation Type 30, but designed and constructed the remainder of the equipment in the Electronic Systems Laboratory. The special modifications in the new display,

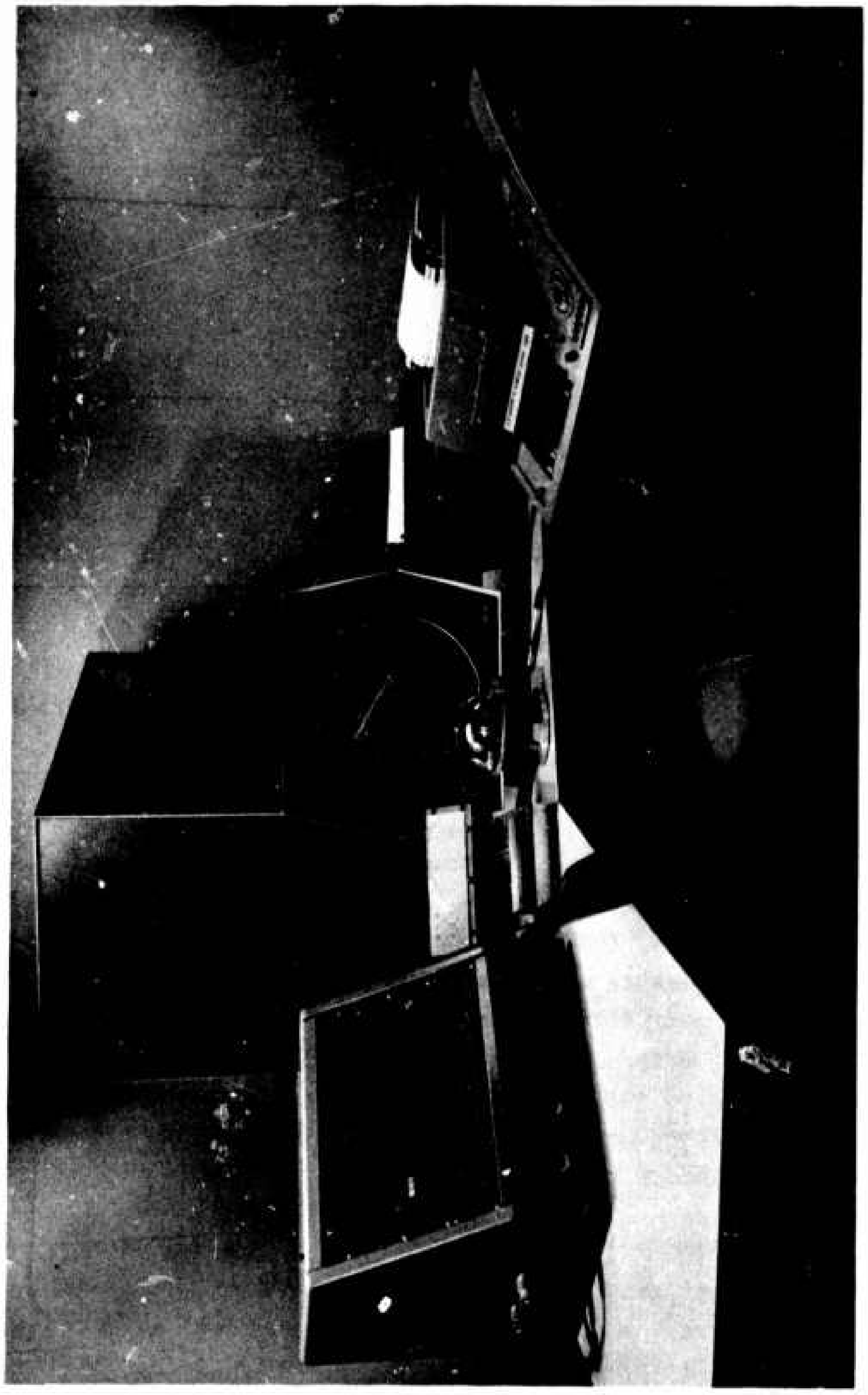


Figure 5. The ESL Display Console

designated Type 330 by DEC, permit incremental as well as whole number inputs to the scope buffer registers. The incremental digital computational logic contained in the ESL-constructed portions of the console provides the incremental inputs to the scope buffers and permits high-speed generation, scaling, translation, and rotation of geometrical figures with a minimum of computer attention.

The console is connected to the Direct Data Device on Channel D. All data flow from the 7094 memory, the A-Core, is in the channel mode in which the console steals a 7094 memory cycle whenever it needs a new data word. Each data word contains a command field which is interpreted in the console. The attention of the 7094 is required only to set up display lists in the proper A-Core buffers, to prepare the channel for each repetition of the display, and to monitor manual inputs from the console.

The console has now been operating in the MAC time-sharing system for six months. A number of research groups in the Departments of Electrical Engineering, Mechanical Engineering, Civil Engineering, and Naval Architecture and Marine Engineering have been using it in projects such as graphic input programs of the Sketchpad type, coordinate geometry problems, and surface fairing for aircraft and ship structures. Because of the novelty and incompleteness of the system, most current uses are of an experimental and exploratory nature. The expansion of display system capabilities and the preparation of more powerful display programming languages are progressing, and the console should soon become a standard tool for these and other research groups.

At present the console electronics and associated portions of the time sharing supervisor program produce an essentially flicker-free display of up to 300 inches of line drawing or 300 alphanumeric characters or some combination of both at the expense of only a few percent of the memory cycles of the project MAC computer. Display programs are run as part of the normal time sharing system without interference to other users. Picture manipulations, such as rotation, translation, and scaling, are performed in real time under control of the light pen, switches, knobs driving shaft encoders, and a unique three-axis globe. Light pen tracking, which in the past required about ten percent of computer time when programmed as a subroutine, is now a completely automatic hardware function. Although it uses essentially no computer time, it still uses ten percent of display time, in part because the light pen tracking circuits cannot use the high plotting speed, 1.5 microsec. per point, of the display system with existing light pens. Time lags in the decay of phosphor intensity and of the photocell response prevent reliable light pen response for points plotted at intervals of less than about 8 microsec. During the past year, we have developed a new type of pen which uses capacitive pickup of the charge created by the electron beam in the display tube. This new beam pen eliminates time-lag problems; and, its greatly improved response speed will soon reduce tracking time.

The Display Group is also concerned with the problem of multiple remote displays for Project MAC in order to provide the capacity for alphanumeric and graphical display at every MAC terminal. We have two immediate problems: the transmission of picture information without special high-bandwidth cables, and the provision of low cost local picture maintenance. In the former problem we are considering the use of either existing telephone services or a broadcast microwave link. For local picture storage, although we are examining other techniques, we are investigating primarily the direct-view storage tube.

B. Progress

1. Hardware Pen Track

Pen tracking was made an automatic function within the console by adding a 20-bit pen tracking register that holds the x and y pen coordinates between tracking cycles. To initiate tracking, the computer first issues on the channel a pen track starting command which contains the x,y starting location and then causes the pen track cross, a special mode of line generator, to appear at the designated location. Every ten milliseconds thereafter the console stops the display generation circuits, swaps the current display location with the old pen track location, and performs a tracking cycle. At the conclusion of the tracking cycle, the new pen coordinates contained in the display registers exchange with the display coordinates held temporarily in the pen track registers and the display process resumes at the point of interruption. Once started, pen tracking is a completely off-line operation, and pen coordinates may be read by the computer at any time, except during a tracking cycle. The computer will usually be programmed to read the pen track register at the beginning of each display cycle.

2. Three Dimensional Rotation

Because of the novel nature of the display generation and rotation system, we constructed the console with initially only two-dimensional capabilities. We used one binary rate multiplier (BRM) pair to generate x and y line components and two BRM pairs to rotate these components about the axis normal to the scope face. During this reported period we added a third register to the BRM line generator to handle z line components and also added a corresponding BRM pair to the rotation matrix. These modifications permit generation and three-dimensional rotation of figures composed of straight lines. Our experiments with three-dimensional rotations of wire-frame figures and warped surfaces have been successful, and they produce a striking visual effect.

We shall continue study on the implementation of a curvilinear generator with digital differential analyzer techniques, which we are also considering for replacement of the BRM's in the rotation matrix.

3. Character Generator

The console originally contained only the ESL-constructed programmable character generator, which uses the BRM line generator to step the scope beam through a 5 X 7 matrix. Thirty five bits in a word supplied by the computer determine the points to be intensified in this matrix. At 1.8 microsec. per point, this character generation requires 70 microsec. per character and one full computer word per character.

Project MAC acquired a Straza stored-character generator, which produces better looking characters than the 5 X 7 matrix. The 64 Straza characters, selected by six-bit codes, are formed by up to 16 points on a 15 X 16 matrix. Character generation time, which is adjustable, can be as fast as 10 microsec. per character but is currently 60 microsec. because of deflection speed problems. We plan to speed up the character display with a dual deflection cathode ray tube which retains magnetic deflection for the main display functions but uses electostatic deflection for the characters. We have used the Straza character generator to label graphs and figures and to display program listings for on-line debugging purposes.

4. Beam Pen

The Display Group has been developing a new type of electrostatic beam pen for a faster response than that of available light pens. This pen avoids the time lags in the phosphor and light detector by sensing the electron beam directly, rather than the light which the beam creates by striking the phosphor. The superposition of a ten megacycle signal on the intensification pulses permits a-c

12

amplification of the beam signal as picked up with a capacitive probe. We have packaged our prototype design, shown in Figure 6, as an active probe with a nuvistor triode as a cathode follower, in a pen shaped holder only slightly larger than that used for light pens. The output signal from the beam pen is practically identical with the intensification pulse. Its field of view is slightly larger than that of most light pens, but we feel that we can reduce this in future designs. We have achieved satisfactory pen tracking and are now installing the prototype pen in the console for experimental use.

5. Remote Displays

After having investigated several modes of local console storage, we selected direct-view storage tubes as the most promising for early experiments. We are now conducting two experiments with a Tektronix Type 564 Laboratory storage scope: one will determine whether deflection signals from the ESL Console can be transmitted within the MAC building by coaxial cable; the other, whether slowed-down deflection signals can be transmitted over telephone circuits with analog data sets.

In the first experiment, with three 200-foot coaxial cables for x and y deflection and for the intensification signals strung from the ESL Console to the remote display, we have displayed satisfactory line drawings and alphanumeric messages in the storage mode. Two such storage scopes will soon be available to MAC users in the public console area. In the second experiment, we have installed six telephone circuits between the ESL Console Room and the MIT switchboard. These lines provide two round trip analog circuits for the transmission of x and y deflection signals (Type 602B Dataphones) and one round trip digital circuit (Type 202B Dataphone) for the transmission of the intensification signals. We expect that the noise and signal linearity characteristics of the analog data sets may

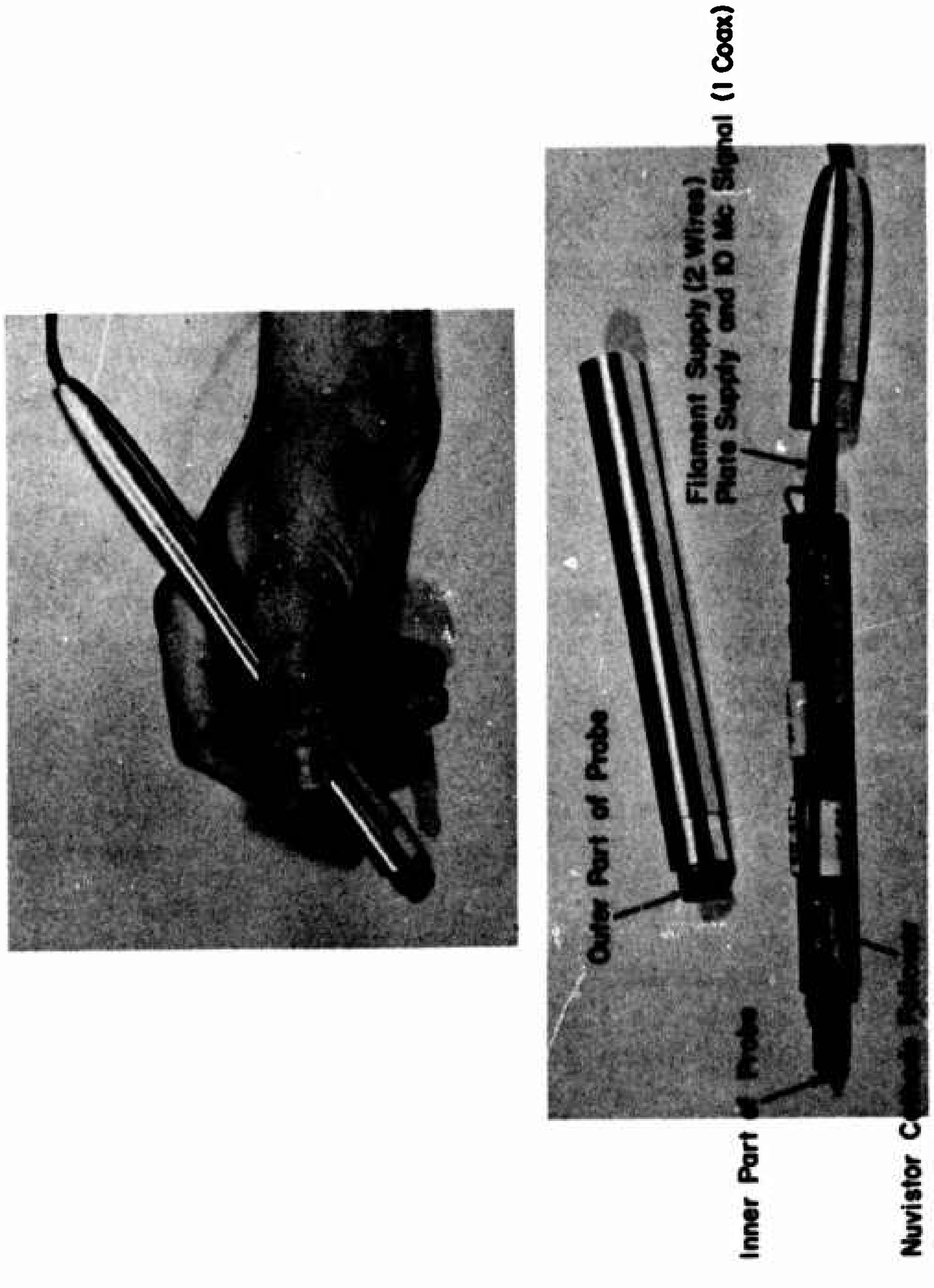

Filament Supply (2 Wires)
Plate Supply and 10 Mc Signal (1 Coax)
Outer Part of Probe
Inner Part of Probe
Nuvistor Cathode Follower

Figure 6. The Electrostatic Beam Pen

be the limiting factors in this mode of picture transmission. The measurement of the characteristics is the goal of this experiment. Should we achieve satisfactory picture transmission, we could quickly make available a few remote displays of this type at selected points of the MIT campus.

6. Second Operator Station

During this reported period a second operator station, which consisted of a display tube and a complete second set of manual inputs was added to the ESL Console by Project MAC. The second display (DEC Type 343) contains only deflection amplifiers, driven in parallel with the deflection amplifiers of the original console display. Since the computer can control which of the two display tubes is to be intensified at any time, the second operator station can be used completely independently of the first station; and, thus, two operators will be able to work simultaneously on different problems. The total flicker-free display generation capability must be shared between the two stations, but the present capability is adequate for two stations. Alternatively, the second station can be used as a true slave display for purposes of scope photography, demonstrations, etc.

7. Display Programming

Since, at present, the ESL console has no memory of its own and must have continuous access to data words stored in the computer memory, such storage should be in the A-core memory which holds the supervisor program, and not in the B-core user memory. For the maintenance of some sort of on-line interaction with the display, each human action should not be subject to the time-sharing delays associated with the access to a user program. Thus, the supervisor program must not only buffer display inputs for later action by a B-core user program but must also perform some functions in real time.

R. U. Bayles wrote an A-core program, called the Console I/O Adapter, to handle these input/output requirements. As part of the time-sharing supervisor, this program occupies 4,000 registers, half of which consist of buffer space for output display lists. For output the adapter program accepts display messages from the user program, stores them in its output buffer, and transmits them to the console 30 times a second under control of a console alarm clock. Input messages, such as seeing a line by the light pen or the pushing of a button, are placed in an input attention buffer for transmission to the user program. Each such input storage is indicated on the scope by a small mark, called a glitch, which is erased when the user's B-core program reads the attention buffer. Thus the user can tell by the glitches whether a particular action was noted by the A-core program and can also readily see when it is passed on to his own program.

In addition to buffering input messages, the I/O Adapter performs some functions directly that permit the drawing and manipulation of figures in real time. One of these is the drawing of rubber-band lines in the following fashion: under the control of a push button, the I/O Adapter initiates light-pen tracking; one push of a draw button then fixes a point at the present pen position and causes the I/O Adapter to draw a straight line between this point and the tracking cross as it is moved about; a second push of the draw button will fix this line and start a new one attached to its end, etc. The coordinates of these lines are set up in an input buffer and held there until they are read by the user program. Moreover, lines held in the input buffer are displayed by the input section of the I/O Adapter and are marked with glitches. When the user program has read the input buffer, the buffer is cleared, and the lines contained in it will no longer be displayed unless the user program transmits them as an output display message to the I/O Adapter. Thus the user may draw continuously with part of his picture in the output state and part in the input state, as indicated by glitches. The only limitation on his speed of drawing is the amount of input information which can be buffered between active periods of the user program.

Another real-time function of the I/O Adapter is to position, scale, and rotate figures under control of the console knobs and the globe control. When instructed by a call from the user program the I/O Adapter reads the input devices; that is, the light pen, buttons, or keyboard, each display cycle (30 times a second), and calculates new values for the three words in the specified display list headings which control the initial h,v location and the rotation matrix settings for that picture element. Thus the user has real time control of the position, size, and rotation of the selected picture element, even though he is using a time-shared system.

Although the I/O Adapter program has convenient calling sequences for transmitting messages to and from user programs, these messages must be written in the console command language. To facilitate user communication with the display, we have written a B-core programming system which consists of a set of subroutines performing standard functions that are read into B-core with the user's program. The B-core system's main facilities are:

1. A set of standard subroutines or procedures whose functions each user would have to provide himself if he were to program the console by using the A-core I/O Adapter directly. When the parameters of a particular part are given to the appropriate procedure, the required console commands are automatically built and added to the display file.
2. Arbitrary names may be given to pictures and picture parts or to any consecutive group of console commands at the time they are added to the display file. Whenever the light pen sees them, both the given name

of that picture and of the particular part of that picture that the pen saw are made available. Then, if the user wants to modify a picture or part of a picture, he may identify it by name.

3. The full A-core system interface is still available to the user if he wants.

Computer-Aided Design Project - Douglas T. Ross

The Computer-Aided Design Project, sponsored by the U.S. Air Force, was begun over five years ago as a cooperative program between the Computer Applications Group of the Electronic Systems Laboratory and the Design Division of the Mechanical Engineering Department. The aim of the project is the application of a computer system to any area of modern design, engineering, and manufacturing so that work in these areas may be carried out more efficiently, more economically, and in greater detail than has heretofore been possible. With the added support of Project MAC, the broad scope of the Computer-Aided Design Project is broadened still further to include general problem solving by man and machine in even wider fields. The Computer-Aided Design system must be general purpose, natural to use, inherently evolutionary, efficient, and essentially independent of the machine. Lack of attention to any of these areas can compromise the whole problem-solving idea which is sought. The Computer-Aided Design Project includes four principal areas of activity:

1. Theory
2. Computer Applications
3. Design Studies
4. Hardware Developments

The latter two areas are described in separate sections of this Progress Report. Herein we describe the major developments in computer applications and theory in the first half of 1964.

In our initial work, we formed a unified viewpoint on representing the structural and dynamically changing quanta of data from which problems are composed. Our viewpoint is that problems can be modelled in terms of elements that contain varying numbers of components. These components, in turn, may contain numerically coded data or pointers to other components so that both the values and interrelationships of data or properties which compose the problem are explicitly exhibited. The resulting network-like structure is called a plex. Plexes combine into larger plexes and may model either data or processes. In addition to their own structures, plexes also suggest the dynamic growth of larger structures from smaller ones. While we have only partially realized the implication of the plex concept, we have already used many applications of them within the Project for the past four years.

One fruitful application has been in the field of language processing. We have written algorithms which specify how the words of an arbitrary language interact first to form phrases, sentences, and then to build up a first-pass structure which models a statement made in the language. The context of each word is shown by a tree structure; and the sequence whereby words seem to build up the meaning of the statement, from the meaning on its subparts, is shown by the precedence-string chain of pointers which threads through the tree. Operator algorithms may then follow the precedence string and transform the meaning of the words taken in context into a modelling plex of that problem which the statement concerns. In this way the detailed complexity of the modelling of a problem arises automatically from convenient language descriptions.

The language-theory algorithms have thus been combined with operators to create a compiler system in which the final modelling plex is an operating machine-code computer program. An initial version of the first such compiler, called AED-0, was completed in late 1963 and incorporated as a user program in the Project MAC Compatible Time-Sharing System. The language of AED-0 is Algol-60, with a few features omitted that are not needed for systems programming and many needed features added for plex programming. AED-0 serves as the programming vehicle for the more general AED-1 system, which will evolve through successive states toward the full concept of Computer-Aided Design.

We are now making a number of major additions to the AED-0 system to provide a more powerful programming vehicle for the AED-1 system. The additions assist the basic operations of writing and testing large programming systems. Among the new facilities is a general and flexible Input-String Macro facility which allows the substitution of strings of characters in other strings, arbitrarily, upon command. Since the form of the macro call is identical to that of the procedure call for the AED-0 compiler, we may mechanize operations interchangeably as either procedures or macros; and, since the macro facility applies directly to uninterpreted input-string characters, we may apply the powerful macro capabilities to any language of sublanguage used in the system.

We are also adding a flexible Breakpoint and Patch facility which permits both source language debugging and flexible program operation. A special mode of the AED-0 compiler will insert breakpoint locations in a compiled program in such a way that arbitrary programs may by spliced into these breakpoints and activated on command without recompilation. As is characteristic of AED programming, instead of a large source-language debugging system, a few new words to be added to the vocabulary will, in effect, allow one program to be used as data for another program. Thus, the debugging language is the compiler language itself and has complete control over all aspects of the processing. Full logical expressiveness of the compiler language is available for debugging. In addition, the compiler itself is being made to operate recursively, so that compilations within compilations will be possible. This revision feature, if combined with an on-the-fly execute feature, will allow very elaborate control over all other systems features.

Finally, preparations for AED-1 programming include a collection of commands for the time-sharing system that transmit files, with both automatic record-keeping and track clean-up from one programmer to another through the common file. When many programmers are simultaneously modifying a large program, a change effected in a portion of the system by one programmer must be brought to the attention of all the others working on the program by means of automatic administration of program handling. Several stages of rigorous routine check-up are also required before new system features may become part of the public command system. These important aspects of project administration and intramural communication are carried out automatically by commands which not only make the work easier, but greatly increase the reliability and responsiveness of the group work. We have made these commands available to other users of the Project MAC system, and are including almost all of the compiler features of the AED Project in the public subroutine library, so that they may be used independently with other systems as well.

Besides its activities in compiler writing, our Computer-Applications Group carries on activities in general language processing. They have developed compact and general versions of the basic First-Pass Algorithm of the language theory, such that the whole project has a richer linguistic foundation. This form of language processing places no restriction on the physical form of the symbols that are used for words in a language, so that we can process not only verbal language, but also graphical or pictorial language as well.

Input/Output Equipment for the PDP-1 - Robert H. Stotz

The Electronic Systems Laboratory has undertaken several projects to provide special input/output equipment for the PDP-1; they include the installation of a 202B Dataphone connection (1200 bits per second) to the 7094, the connection of a specially adapted Siemens five-level teletype machine to provide Braille output, and study of the mode of connection of the PDP-1 to the 7094 and to the ESL Display Console. In order to provide a flexible and easy means of adding these and other I/O devices to the PDP-1, we designed a special I/O Box and had it built for us by DEC. This unit has provisions for decoding up to 512 iot commands, an expanded input mixer with taper pin terminal blocks, and two eighteen bit output buffers.

We installed the Dataphone early this spring; but, since the 7750 and 7094 software package necessary to incorporate it into the CTSS was not completed we have not yet been able to use it. The Braille teletype, which is on loan from Mr. Calvin Mooers, Rockford Research Institute, Inc., contains a keyboard for input and a punched paper tape output, with two of the five punches backed off so that they do not strike the paper and the three least significant bit punches adjusted so that they indent the tape rather than create fully punched holes. This arrangement can generate a Braille output character (2 X 3 matrix) with the output of two coded characters followed by a space. The PDP-1 output command is iot x015; and the five output bits are the five least significant bits in the I/O Register. Bit 10 of the

Check Status word is tested to determine the availability of the machine for output. Striking a key sets bit 11 of the Check Status word and causes a sequence break. Execution of iot x115 then clears the I/O register and loads it with the code associated with the key. Since this input operation produces the entire five bit teletype code, which has no relation to the Braille characters, the keyboard input goes only to the computer, and does not operate the punch.

Our investigation on a Direct Data Interface to the 7094 and the Display Console proceeded to the point of a logic design, but the decision to replace the PDP-1 with a PDP-6 made this study obsolete. We shall shortly determine a mode of connection of the PDP-6.

LINCOLN LABORATORY

A Computer Technique for Optimization

A Method for On-Line Manipulation of Data Files

A Computer Technique for Optimization - Homer C. Peterson

Our computational problem involves the optimization of a function for which the mathematical definition of optimal is either difficult or unknown. Specifically, we are trying to optimize a unified S-band carrier which contains telemetry data, ranging code, and voice so that we may readily separate the information. We first found a program that gave a representation of this model in a time short enough to allow effective on-line monitoring of the computations. The delays in the system originally were thirty to forty five minutes between the entry of a set of parameters and a solution. We next devised a crude teletype plotting technique and evaluation method to check the reliability of the model and to give a rough presentation of the data.

In our current research we have isolated certain parts of the data and presented this information to the user in numerical form from which he could tell whether a change in the parameters improves the output. When the program finds a numerically better set, it generates a rough plot; and, if this plot does not reveal any glaring faults, a background program generates a finer detailed plot. In general, what we have developed is a technique for optimization of a mathematical model in which both the equation is of such a nature as to require computer solution and its mathematical analysis for optimization is unknown. In the absence of time-sharing the long turnaround and the many unnecessary tries inherent in submitted runs and subsequent evaluation would make this technique impractical. Since our model was an equation which involved transcendental functions and infinite sums, an exact program representation of it was impossible; however, on-line operations were advantageous to us in checking our programmed model against test cases.

A Method for On-Line Manipulation of Data Files - John Nolan

The aim of this research program is a method for on-line manipulation of data files of formated data. Versatile data storage and retrieval systems will play a critical role in time-sharing of computer facilities. In such diverse applications as technical data libraries, automated teaching systems, management information systems, or military command/control, extensive files of data must be readily available to system users. In the past several months we have designed a general purpose data storage, retrieval, and editing system for on-line use, which has the following features:

1. It tells the user what information it needs to set up files, informs him of input errors, provides him with information about the files, and gives him a list of choices available for input operations.
2. The files that the user constructs can vary in length, organization, and content.
3. The system sets up all formats. The user refers to the file contents, such as data values, by name only; and, thus, he does not need to know the specific physical structure of the files by name.
4. The user may redefine or modify files on-line with ordinary English commands and may call the files, or parts of files by name.
5. The user can create many files that differ in length and content and establish relationships between files and parts of files. He can thus organize the same set of data under a number of category headings or data from a number of different files under a single category heading. The system automatically carries out this multiple association of data sets by means of link structures; duplicate storage of data is not necessary.

Conventional data storage and retrieval systems for special applications are designed to operate upon files of fixed format with a fixed set of user queries. For greater versatility of operation more recent systems store selected information internally in master files on the formats of the files and thereby provide response to a broader class of user queries. The system under design allows the user to modify these master files on-line and to change both the file contents and data retrieval operations. Changes to master files result whenever we define new files, modify or delete old file definitions, define new cross-associations of existing files, and so on. The user will also be able to request of the system special formats for magnetic tape records of data, which will furnish an automatic translation of the data into and from the system's files. These features allow multiple users to edit and select data from common files for special use.

The system design stresses versatility; it is intended for many different classes of users and of stored data. A principle design problem was an encoding scheme for internal descriptions of file organizations defined by the users. The scheme exploits a simple model, applicable to all data files, in which a multilevel hierarchial tree of replicated subfiles and/or data fields represents file organization. Despite differences in data content and meaning, the same model applies to files of formatted data in any application.

The technique for encoding the file descriptions of systems has several unique features:

1. Files are capable of free cross-association; any file's entries may be linked as a subfile of any other file's entries.

2. Any file may simultaneously be cross-associated by means of any number of relations with other files, including itself.

3. The physical location of data files or links within entries or the location of entries within the avalble memory are under the control of the master files in a wholly variable fashion and without the knowledge of the users.

4. The master files themselves are considered part of the data files, hence, they also completely describe their own organization. With this feature the system designers may modify the master file definitions for boot strapping purposes with the same facilities available to general users.

We have programmed an initial model of this data storage and retrieval system, and it is now under test on CTSS from remote teletypes at Lincoln Laboratory.

This system is being used to experiment with on-line manupulation of data files. To test the versatility of the system we shall introduce sample files which represent a variety of classes of data such as numerical data files, administrative data files, and bibliographic reference data. A general purpose routine has been written for the description of on-line formats of punched card files and for the reading and storage of such card file images from tape. These routines will provide a broad class of sample files for experimental testing.

APPENDICES

Appendix A
Publications Other Than MAC Technical Reports

Appendix B
Project MAC Memoranda

Publications Other Than MAC Technical Reports

Bers, A. and Briggs, R.J. "Criteria for Determining Absolute Instabilities and Distinguishing between Amplifying and Evanescent Waves." Bull. Amer. Phys. Soc., Series II, Vol. 9, No. 3. March, 1964.

Bers, A. and Mills, J.D. Computer Analysis and Display of Wave Instabilities. M.I.T., R.L.E. Quart. Prog. Report No. 74. July, 1964.

Bobrow, Daniel G. and Raphael, Bertram. " A Comparison of List-Processing Computer Languages: Including a Detailed Comparison of COMIT, IPL-V, LISP 1.5, and SLIP." Commun. of ACM, Vol. 7, No. 4. April, 1964.

Bobrow, Daniel G. "A Question-Answering System for High School Algebra Word Problems." Proc. Fall Joint Computer Conference. 1964.

Brach, John and Chamby, J.A. The Use of Linear Programming in Architectural Design: The space Allocation Problem. M.I.T. Civil Eng. Systems Division. May, 1964.

Dailey, James E. A Digital Computer Simulation of Unsteady Flow in Rectangular Non-Prismatic Channels. M.I.T., Dept. of Civil Eng. B.S. and M.S.

Dennis, Jack. " Use of Computers in Speech Research. Computers in Medicine and Biology. Ann. New York Acad. Science, 1964.

Electronic Systems Laboratory of M.I.T. Annual Report from July 1, 1963 to June 20, 1964.

Everest, Gordon and Selwyn, Lee L. Considerations for an Interactive Algebraic Compiler. M.I.T., Sloan School of Management. June, 1964.

Fenves, S.J., Logcher, R.D., and Reinschmidt, D.F. Preliminary Version STRESS: A Problem-Oriented Language for Structural Engineering. M.I.T., Dept. of Civil Eng. Technical Report T63-8. Oct., 1963.

Fenves, S.J. Logcher, R.D., Mauch, S.P., and Reinschmidt, D.F. STRESS: A User's Manual, a Problem-Oriented Language for Structural Engineers. M.I.T. Press. 1964.

Gladdings, Dale G. Automatic Selection of Horizontal Alignments in Highway Location. M.I.T., Dept. of Civil Eng. M.S., May, 1964.

Greenberger, Martin. "The Computers of Tomorrow." Atlantic Monthly. May, 1964.

Graham, Robert M. " Bounded Context Translation." Proc. Spring Joint Computer Conf. 1964.

Kessler, A.R. Structural Synthesis and Analysis, and Simulation of Interpersonal Power Systems. M.I.T., Dept. of Economics and Social Science B.S. June, 1964.

Krakauer, Lawrence J. Syntax and Display of Printed Format Mathematical Formulae. M.I.T. Dept. of Elect. Eng. M.S. June, 1964.

Linderman, James L. The Role of Memory in Information Storage and Retrieval. M.I.T., Dept. of Math. B.S. June, 1964.

Manheim, M.L. Highway Route Location as a Hierarchically Structured Sequential Decision Process: an Experiment in the Use of Bayesian Decision Theory for Guiding an Engineering Process. M.I.T. Dept. of Civil Eng. Ph.D. May, 1964. Civil Eng. Research Report R64-15.

Minsky, M.L. and Cocke, J. "A Universality of Tag Systems with P=2." J. ACM, Vol. 11, No. 1. Jan., 1964.

Papert, S. Stereoscopic Synthesis as a Technique for Localizing Visual Mechanisms. M.I.T., R.L.E. Quart. Prog. Report No. 73. April, 1964.

Papert, S. Regularities in the Time Courses of Some Visual Processes. M.I.T., R.L.E. Quart. Prog. Report No. 73. April, 1964.

Puri, Statish. Electron Beam Interaction with Ions in a Warm Electron Plasma. M.I.T., Dept. of Elec. Eng. M.S. April, 1964.

Raphael, Bertram. "A Computer which Understands." Fall Joint Computer Conf. 1964.

Roos, Daniel and Miller, C.L. COGO-90: Engineering User's Manual. M.I.T., Dept. of Civil Eng. Res. Report R64-12. April, 1964.

Roos, Daniel and Miller, C.L. The Internal Structure of COGO-90. M.I.T., Dept. of Civil Eng. Res. Report R64-5. Feb., 1964.

Roos, Daniel and Miller, C.L. COGO-90 Time-Sharing Version. M.I.T., Dept. of Civil Eng. Res. Report R64-18. May, 1964.

Selwyn, Lee L. _A Computer-Controlled Tele-Communication System._ M.I.T., Dept. of Industrial Management . 1964.

Smith, A.A., Slosberg, D., and Van Horn, E.C. Jr. _The Scientific Translating System._ M.I.T., R.L.E. Quart. Prog. Report No. 73. April, 1964.

Teager, Herbert. "Mass Memory Requirements for On-Line Real-Time Systems." _Spring Joint Computer Conf._ April, 1964.

Teager, Herbert,"Workshop on Problem Solving through the Use of Automatic Displays." _Fall Joint Computer Conf._ 1964.

Teitelman, Warren. "Real Time Recognition of Hand-Drawn Characters." _Proc. Fall Joint Computer Conf._ 1964.

Project MAC Memoranda

Memorandum MAC-M-NO.	SUBJECT	AUTHOR
20	Cybernetics and Technical Progress	A. Berg
21	Kiev Cybernetics Section	L.A. Kaluzhinin A.A. Stovniy
22	Future of Cybernetics	V. Gukov
23	Topics for Discussion at Meeting on Intergalactic Computer Network	J.C.R Licklider
24	Time-Sharing: Justification, Problems, Solutions	L.L. Lombardi
25	Notes on Time-Sharing	N.H. Taylor
26	Reactive Typewriter Meeting	F.J. Corbato
27	Man-Machine Relations	N.H. Taylor
28	Problem Areas of Time-Shared Computer Systems	F.J. Corbató
29	Problems on Synthesis of Decoding Trees	W.H. Kautz
30	Thoughts on Project MAC	M.V. Wilkes
31	Non-Optical Graphical Input for CRT Displays	D.C. Englebart

32	Time-Sharing Computing Systems	J. McCarthy
33	The Octopus Report	N. Hardy
34	Outline of a Large Computer System	L. Monrad Krohn
35	Making the Scientist More Effective	D. Engelbart
36	Thoughts on Machine-Aided Design System for Multi-Queue	R.L. Sisson
37	Province of Panel on Users	D.B. Yntema
38	Telephone De-bugging	N.H. Rochester
39	Direct Communication Between Independent Business Computation Centers - a cover for three articles:	L.L. Lombardi
	System Communications	S.K. Plotnick
	Computer Communication - A Problem That Can Be Solved	S.K. Plotnick
	Logic of Automation of System Communications	L.L. Lombardi
	Some Thoughts with Respect Meeting the Technical Objectives of Project MAC	H.M. Teager
41	Thoughts on Users and Terminals	J.A. Rusling
42	On Aiding Creativity	U. Neisser

43	Project MAC and the University Environment	R.M. Brown
44	Functions Desirable in Time-Sharing System	JCR Licklider
45	Group Interaction, Confernclng and Consulting in the Time-Sharing System	A. Rosenberg
46	Some Audio I/O Possibilities for MAC Systems	G.A. Miller
47	Common and Uncommon Requirements Between Users	N.H. Taylor
48	Thoughts on Users and on (at least one) Current Compiler Language	K. Smith
49	Terminals in First Phase of MAC, Remarks on User Requirements	D.B. Yntema
50	Brief Functional Description of ESL Display Console	J.E. Ward
51	Modus-Operandi: A Suggestion	F.F. Lee
52	Simulation and Dynamic Modeling	M. Greenberger
53	Self-Teaching Program On and Through Time-Sharing	D. Eastwood
54	MEMEX 1963	R.F. Simmons

55	List of Users' Requirement Factors: Version 1	R.L. Sisson
56	Location and Distribution of Terminals	A.I. Dumey
57	Thoughts on the Philosophy of MAC	R.L. Sisson
58	Manipulation of Tables of Psychologic	D.B. Yntema
59	System Planning	E.S. Fergusson
60	On-Line Control of Experimental Procedures	J.I. Elkind
61	Functional Specification for Languages and Compilers	J. Sammet
62	Notes on the MAC Schedule	R.L. Sisson
63	An Experiment	A.I. Dumey
64	Typewriter Features Considered Desirable	JCR Licklider
65	Notes on the "Structure" of Computer Technology	R.L. Sisson
66	Facilities (Users' Language) Phase 2	R.L. Sisson
67	Parallel Consoles	U. Neisser
68	A Proposed Terminal System	F.J. Corbato

	for the Project MAC and Computation Center IBM 7094 Computers	
69	Topics Relevant to Administrative Policy	A.A. Cohen A.I. Dumey J.A. Rusling
70	Terminals: Definition, Problem Areas and Some Suggested Activities	F.F. Lee
71	What are We Trying to Do?	R. Kain
72	Project Summer Study, 1964	D.K. Pollock
73	Answer to JCR Licklider's Report of August 5	R. De Vogelaere
74	Short Thoughts on Long Memories for Language Processing	R.F. Simmons
75	Skeleton of Part of Users Panel 1 Report	D.B. Yntema
76	Command Language	N.H. Rochester
77	Project MAC and the "Small User"	E.S. Fergusson U. Neisser
78	Steps on the Road to Memex	U. Neisser
79	A Comment on MAC Planning	W. Jocobs
80	Evaluations of Importance to Users	D.B. Yntema

81	MAC Planning	N.H. Hardy
82	Tentative Teletype Equipment Order	R.G. Nills
83	Learning from Users	J . Weizenbaum
84	Tentative Outline for Final Report	J.B. Dennis
85	The Naive User	K. Smith
86	Criteria for Using a Display System as an Input-Output Device	N. H. Taylor
87	Draft of Section on Teaching Routines in the Light of Panel Discussion	U. Neisser
88	Item Identification in Dialogues or, "Are We Talking About the Same Thing"	R. M. Brown
89	Programming Systems for the MAC Time-Sharing System	M. L. Minsky
90	GGEN: A MAD Language Generation System for Timesharing	R. F. Simmons
91	Summary of Comments on Administrative Ploicy	A. A. Cohen A. I. Dumey J. A. Rusling R. L. Sisson

92	Five Weeks Experience with CTSS (Yntema's A.1.f.)	CTSS Users (K. Smith, Secy.)
93	Surveillance of Technological Progress	A. A. Cohen
94	Results in an Experiment in Man-Machine Reaction	D. C. Engelbart
95	Longer Range Requirements of Student Users	R. De Vogelaere
96	A Form of Group Interaction Easily Available on CTSS	D. C. Engelbart
97	A Brief Description of the Machine Structure Proposed by Amdahl	R. M. Brown
98	A Viewpoint on MAC Research	D. C. Engelbart
99	Look Ahead Memory Implementation	F. F. Lee
100	General Needs of the Scientific User	D. McCowan
101	A Multiprogramming Multi-processing Simulator	N. H. Rochester
102	Console Central and the Tea Cart Console	E. S. Fergusson
103	An Evaluation of the Compatible Time-Sharing System	M. V. Wilkes

104	Incremental Data Assimulation In Open-Ended Manned Computer Systems	L. Lombardi
105	User Functions and User Groups	N. Taylor and G. Culler
106	Systems Organization	N. Taylor
107	A Proposal for Modifying the IBM 7094 Computer for a Directly Addressable Large Core Memory	F. J. Corbato
108	Project MAC - It's Goals and Status	R. G. Mills
109	Revised Plans for PDP-1 Input-Output	J. E. Ward
110	Structural Description of a Modular Time-Sharing System	Daley, Corbato Daggett, Hellwig
111	Operating Procedure for the Interactive Version of 1620 FORGO	L. L. Selwyn
112	Fast-Interaction Around-the-Clock Remote Consoles	M. L. Minsky
113	How to Break the Log-Jam on Multiple Display Consoles	M. L. Minsky
114	Proposal for the 7094 Direct Data Channel Switch	F. F. Lee

115	An Approach to a Dynamic Storage Allocation	E. Van Horn
116	Fortran Improvements	L. Selwyn
117	Tailored User Terminals	S. Ferguson
118	A Proposal for a Mathematical Manipulation-Display System	M. L. Minsky
120	Survey of the Requirements for an M.I.T. Interactive Algebraic Compiler	Colman, Everest, Lombardi, and Selwyn
121	Time-Shared Computer Scheduling	R. Y. Kain
122	Demon: ESL Display Console Demonstration Program	R. B. Polansky
123	Real Time Serial Computation	D. Arden
124	Derivator I	M. L. Minsky
125	ESL Display Console Time Studies	
126	A Proposed Improvement of the 7094 Memory Protection RPQ	Corbato
127	Further Plans for a PDP-1 Input-Output Buffer Unit	J. E. Ward

1✓

128	Secondary Storage in LISP	D. Edwards
129	LISP Garbage Collector	M. L. Minsky
130	The I/O Input-Output Adapter in the Modular Time-Sharing System	Creasy, Hellwig, Corbato
131	Look-Aside Memory Simulation	F. Lee
132	LISP System	D. Edwards
133	A Directory to the MAC Time-Sharing System	Hellwig, Corbato
134	LISP Exercises	T. Hart
135	The Graphical Typewriter	M. L. Minsky
136	C.T.S.S. Document Tape	C. C. Staff
137	A Machine Structure for Dynamic Storage Allocation	J. B. Dennis
138	A Proposal for a MAC Standard 1050 Golf Ball	M. I. Levin
139	C.T.S.S. Version "CHAIN"	Logcher
140	On-Line Version of GPSS II	M. M. Jones
141	Specifications for IBM Selectric Input/Output Writer for ESL Display Console	J. Ward

142	LISP As the Language for an Incremental Computer	B. Raphael and L. Lombardi
143	Research Program in Machine Structure	J. B. Dennis
144	Procedure for Obtaining User Comments	R. G. Mills
145	Corrections to the Arden Serial Multiplier	R. G. Mills
146	AED-0 Programmer's Guide	C. G. Feldman D. T. Ross
147	Short Time-Sharing Console Problems	F. J. Corbato
148	Natural Language Input Problem-Solving System	D. G. Bobrow
149	Translating Ordinary Language Into Symbolic Logic	J. L. Darlington
150	Disk Storage Capacity	R. G. Mills
151	Typewriter Communication with a Multi-Access Computer	J. B. Dennis
152	CTSS Technical Notes	J. H. Saltzer
153	Time Sharing LISP	W. A. Martin T. P. Hart

154	Warnings and Restrictions in AED-0	C. G. Feldman
156	COMIT Operation in the MIT Time-Sharing System	V. Yngve
157	Emergency Procedures for Conservation of Disc Storage	R. G. Mills
158	Syntax of the New Language	M.Levin
159	New Language Storage Conditions	M. Levin
160	Recursive Procedures in AED-0	G. Rodriguez
161	Statistics for C.T.S.S.	F. Whitelaw and M. Jones
162	Remote Graphical Display at High Burst Speed	F. F. Lee
163	A Processor Design: MAP-1	
164	Slave Memories and Dynamic Storage Allocation	M. V. Wilkes
165	Hash Coding Functions of a Complex Variable	W. A. Martin

166	B-Core System for Programming the ESL Display Console in Time Sharing	C. Lang
167	A New Editing Program	A. L. Samuel
168	A Subroutine Trace Program for C.T.S.S.	B. Wolman
169	LOADER: A New Version of the BSS Loader	B. Wolman
170	Comments on Slave Memories	R. Y. Kain
171	Problem T113 Conventions for Disk Storage and Program Write-Up	C. G. Feldman D. T. Ross

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Massachusetts Institute of Technology Project MAC	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED 2b. GROUP None
3. REPORT TITLE Project MAC Progress to July 1964	
4. DESCRIPTIVE NOTES (Type of report and inclusive dates) annual progress	
5. AUTHOR(S) (Last name, first name, initial) collection of material from many Project MAC participants; no specific author or authors	
6. REPORT DATE 1 July 1964	7a. TOTAL NO. OF PAGES 174 + xii 7b. NO. OF REFS
8a. CONTRACT OR GRANT NO. Office of Naval Research, Nonr-4102(01) b. PROJECT NO. c. Nr-048-189 d.	9a. ORIGINATOR'S REPORT NUMBER(S) MAC-TR-1 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
10. AVAILABILITY/LIMITATION NOTICES Qualified requesters may obtain copies of this report from DDC.	
11. SUPPLEMENTARY NOTES None	12. SPONSORING MILITARY ACTIVITY Advanced Research Projects Agency 3D-200 Pentagon Washington, D.C. 20301

13. ABSTRACT

The broad goal of Project MAC is experimental investigation of new ways in which direct links to on-line computers can aid people in their individual work; whether research, engineering design, management, or education.

This is the first annual Progress Report describing the research which has been carried out under the sponsorship of Project MAC. The details of this research may be found in the publications listed at the end of the report.

14. KEY WORDS

Computers	On-line computer systems	
Machine-aided cognition	Real-time computer systems	
Multiple-access computers	Time-sharing	Time-shared computer systems

12

ADMINISTRATION AND SERVICES

SCHOOL OF ENGINEERING

CIVIL ENGINEERING DEPARTMENT

RESEARCH LABORATORY OF ELECTRONICS

SCHOOL OF HUMANITIES AND SOCIAL SCIENCE

SLOAN SCHOOL OF MANAGEMENT

SCHOOL OF SCIENCE

COMPUTER SYSTEM RESEARCH

COMPUTER COMMUNICATION STRUCTURES

ARTIFICIAL INTELLIGENCE

LIBRARY RESEARCH

ELECTRONIC SYSTEMS LABORATORY

LINCOLN LABORATORY

TECHNICAL REPORTS PUBLISHED BY PROJECT MAC

REPORT NOS.	ASTIA NOS.	TITLE	AUTHOR(S)	DATE
MAC-TR-1	AD-604-730	Natural Language Input for a Computer Problem Solving Language	Bobrow, D.G.	6/64
MAC-TR-2	AD-608-499	SIR: A Computer Program for Semantic Information Retrieval	Raphael, B.	6/64
MAC-TR-3	AD-608-501	System Requirements for Time-Sharing Computers	Corbató, F.J.	5/64
MAC-TR-4	AD-604-678	Verbal and Graphical Language for the AED System	Ross, D.T. Feldmann, C.G.	5/64
MAC-TR-6	AD-604-679	STRESS: A Problem-Oriented Language for Structural Engineering	Biggs, J.M. Logcher, R.D.	5/64
MAC-TR-7	AD-604-680	OPL-I: An Open Ended Programming System Within CTSS	Weizenbaum, J.	4/64
MAC-TR-8	AD-604-681	The OPS-1 Manual	Greenberger,M.	5/64
MAC-TR-11	AD-608-500	Program Structure in a Multi-Access Computer	Dennis, J.B.	5/64
MAC-TR-12	AD-609-288	The MAC System: A Progress Report	Fano, R.M.	6/64