16.01.004.033

October 2, 1961

IBM CONFIDENTIAL
TECHNICAL REPORT

A SURVEY OF THE REMOTE SCIENTIFIC COMPUTING PROBLEM

L. E. Rickard and F. B. Wood

ABSTRACT

The objective of the remote scientific computing project (RSC) is to bring the capacities and speeds of large IBM computers within reach of every scientist and engineer, at reasonable cost to the user and with reasonable profit for IBM. A marketable RSC system must include (1) low problem-solving cost, (2) low standby cost for intermittent use, and (3) faster access and shorter response time than in present computing systems.

This survey identifies the kinds of users to be served and the kinds of problems to be solved; outlines the general problem of developing an RSC system and itemizes the principal decision points in such development; and suggests appropriate division of responsibility for the completion of the project. Choices regarding source language, remote and central hardware, present and future communication links, and other problem areas require various approaches, some simultaneous and some sequential: theoretical study, simulation, experimental systems, and commercial evaluation. These approaches are matched with the problems listed in order to show the pattern of interdependent effort needed and thus to facilitate cooperation among the several IBM groups involved in developing an RSC system.

In preparation for a complete system study, this preliminary survey evaluates applicable methods and equipment, both available and planned; describes some possible RSC configurations; recommends a full system study coordinated with the marketing of small scientific computers; and, in the Appendix, compares remote scientific computing with the computing services which could be offered to commercial customers by Multiple Business Systems (MBS).

International Business Machines Corporation
Advanced Systems Development Division Laboratory
San Jose, California

This document contains information of a proprietary nature and is classified IBM CONFIDENTIAL. No information contained herein shall be divulged to persons other than IBM employees authorized by the nature of their duties to receive such information or individuals or organizations authorized in writing by the Technical Director of the Advanced Systems Development Division or his duly appointed representative to receive such information.

Lithographed in U.S.A.

CONTENTS

I.	INTRODUCTION	1
II.	GENERAL DESCRIPTION	1
	A. The Objective	2
	B. Users	2
	C. Classes of Problems	3
III.	PRELIMINARY DESCRIPTIVE DATA	5
	A. Process of Problem Solving	5
	B. Languages	7
	C. Hardware	9
IV.	PRIMARY SYSTEM PROBLEMS	9
	A. Reliability, Efficiency and Economics	9
	B. Human-to-Computer Communication	14
	C. Debugging	15
	D. Verification	17
	E. Communication Between Remote and Central	18
	F. Terminal	19
	G. Processor	19
	H. Response Time	22
	I. Engineer Education	23
	J. Management Education	23
	K. Storage of Source Programs	24
V.	POSSIBLE SYSTEM CONFIGURATIONS	24
	A. General Configuration	24
	B. Alternate Configuration - IBM France	28
	C. Other Configurations	28
VI.	RELATED ACTIVITIES	30
VII.	RECOMMENDATIONS	32
VIII.	CONCURRENT SYSTEM STUDY AND MARKETING OF SMALL SCIENTIFIC COMPUTERS	36
ACKNOWLEDGMENT		41
APPENDIX		43
REFERENCES		46

A SURVEY OF THE REMOTE SCIENTIFIC COMPUTING PROBLEM

I. INTRODUCTION

International Business Machines can strengthen its position in the computer field and increase its profits by fulfilling the need for a well-designed remote scientific computing system. An overall increase in the use of computers could very well occur as a by-product of remote scientific computing.

The recent activity of this project has laid the groundwork for a complete system study, and makes it possible, in this report, to define the general remote scientific computing problem and to point out areas which require further system study. The overall problem of remote scientific computing is analyzed here, and sections VII and VIII recommend several approaches to the solution of the various problems described. The remote scientific computing (RSC) problem is compared with the multiple business system (MBS) problem in the appendix.

II. GENERAL DESCRIPTION

The general remote scientific computing problem has been described in part in the formulation of the objective of this project. The problem may be further defined by classifying potential users of an RSC system, and by classifying scientific problems which might be solved by a remote scientific computing system.

A. THE OBJECTIVE

The objective of the RSC project is to develop a system which will bring the capacities and speeds of large IBM computers within the reach of every scientist and engineer. The system must provide users (1) reasonably low standby cost for intermittent use of large IBM computers, (2) low problem-solving cost, and (3) faster access to computers and shorter response time than is possible with present systems. RSC service must obviously be provided at a cost competitive with current costs of comparable service.

Several factors have delayed universal use of large IBM computers by engineers and scientists: distance to and cost of a central computer, reluctance to try unfamiliar techniques, lack of motivation, and lack of training. These hindrances could be overcome in the type of RSC system contemplated. By means of a remote terminal, any user will have direct telephone contact with a central computer operated by his company or rented from a Service Bureau Data-Center. In such a situation, with the handicaps of distance and delay eliminated, the problems of reluctance, lack of motivation, and lack of training can also be solved. The successful remote scientific computing system will be one in which the user feels that his problem is receiving the same attention by the central processor as it would receive if he were sitting at the central console.

B. USERS

Large IBM computers can be made available to engineers and scientists in large, medium, and small companies, in universities and colleges when remote scientific computing systems are established. An RSC system could serve as a service center for a number of small engineering firms, as an educational aid to universities and colleges, and as an "open shop" computer operation for large companies. The users would include those who cannot afford and/or do not require the full time use of a large, high-speed computer; those whose problems do not require a large computer with large storage capacity and/or high speed; those who require a computer only occasionally. These groups represent a considerable potential market for remote scientific computing systems.

An RSC system might serve terminals in a single department, inter-department terminals in a single company, or terminals for several companies. Transmission between the remote station and the central may be established by leased line or dial telephone connections. The latter would be attractive for an intra-company system where the established company telephone network could serve for data transmission as well as voice transmission. The dial telephone system also offers advantages to small companies and universities in remote parts of the country, providing direct access to computing centers over the switched telephone network.

C. CLASSES OF PROBLEMS

In order to determine a marketable remote scientific computing configuration, the various types of scientific problems must be analyzed and grouped by computational requirements and frequency of occurrence. The first step in this study is to analyze a large sample of actual scientific problems in terms of urgency, available packaged programs, amount of input and output data, storage requirements, debugging (time/run, number of runs), overlap of input, output and computing. Then the problems having similar characteristics should be grouped. The computational requirements will then suggest new configurations or make it possible to recognize existing configurations as suitable for solving certain classes of problems. Having established classifications on the basis of a large sample of typical problems, we can then survey a wider range of problems and determine the frequency of occurrence of each class.

Table 1 is a record of data supplied by mathematician-programmers in the San Jose ASD Computation Laboratory. This sample data from actual logs does not necessarily include the types of problems which the engineer or scientist would attempt to solve himself, and these should not be overlooked in the suggested study. Such problems might more frequently be submitted for computer solution if the engineer or scientist had, as proposed, a direct line to computer facilities.

Many applications include more than one set of computational requirements. The first application listed in Table 1, for example, includes at least two general types of problems: transcendental equations and data reduction. The classifications based on similar characteristics will thus be hybrids of problem types, present in various combinations.

CLASSIFICATION OF
(SAMPLE DATA FROM LOGS OF SAM

Type of Problem (and Applications)	Data Requirements (Number of Characters)		
	Input	Output	Storage
Transcendental Equations* Data Reduction (Cam Design)	230	20, 000	60, 000
Data Reduction (Error Detection)	7, 500	1, 500	90, 000
Matrix Inversion Data Reduction Information Storage and Retrieval (Lexical Processing)	800, 000	18, 000	200, 000
Fourier Transforms and Inversions*	2, 100	5, 000	10, 000
Polynomial Roots*	120	240	10, 000
Curve Fitting*			30, 000
Simultaneous Linear Equations*	120	120	12, 000
Numerical Integration*			13, 000
Laplace Inversion*	100	2, 500	10, 000
Ordinary Differential Equations*	150	30, 000	50, 000
Partial Differential Equations	100	200, 000	10^5 up
Roots of Nonlinear Equations	100	200	25, 000
Series Evaluation	50	100	10, 000
Statistical Probability			
Information Processing			
Mathematical Programming			
Simulation			
Algebraic and Transcendental Equations			

: PROBLEMS FOR RSC
ASDD COMPUTATION LABORATORY)

Time Requirements						
Debugging		Production		Overlap of Input/Output and Computing	Urgency	% of Total
es** un	Runs Per Year	Minutes** Per Run	Runs Per Year			
		5	100			
	12	5	1000			
	12	2, 000	3			
		5	50			
		5	30			
		15	10			
		5	10			
		5	10			
		10	150			
	5	5	100			
	20-30	5 up to hours	1000			
	5	1	200			
	5	1	200			

* indicates availability of packaged program
** assumes use of IBM 704

Classifications of scientific problems and problem-solving methods useful in the proposed problem study have been suggested in general studies of the use of digital computers. [1, 2] Marketing surveys already made may furnish further information about the kinds of scientific computing problems which potential IBM customers have.

Once the various problems have been classified according to computational requirements, the tabulated results may be used to determine a remote scientific computing system suitable for present problems under average conditions. The classifications will be useful in determining system variations such as the use of both leased and dial-up telephone lines for transmission of programs and data, the use of optional input-output equipment, and the use of mail service for transmission of high-volume input-output.

III. PRELIMINARY DESCRIPTIVE DATA

Some system study has been made and data collected which describes, in a preliminary way, essential factors in RSC.

A. PROCESS OF PROBLEM SOLVING

To design an optimum RSC system, one must analyze the special process of problem-solving from a remote terminal. Some suggestions from the generalized theory of problem solving may be applied here. Four standard steps for solving problems, as outlined by G. Polya, [3] are listed in Fig. 1, on the left, and are expanded on the right side of the diagram. From "devise a plan" in expanded step two, a choice of one of six paths must be made. In expanded step four, alternate paths are available - one to continue the flow through areas X, Y, and Z, and the other to S. At S alternate paths lead to "redefine problem" (step one) and to W (rewrite). Return to step one is possible (though not shown) from any block. Return to "W" is for correction or modification of the steps in the method of problem solution.

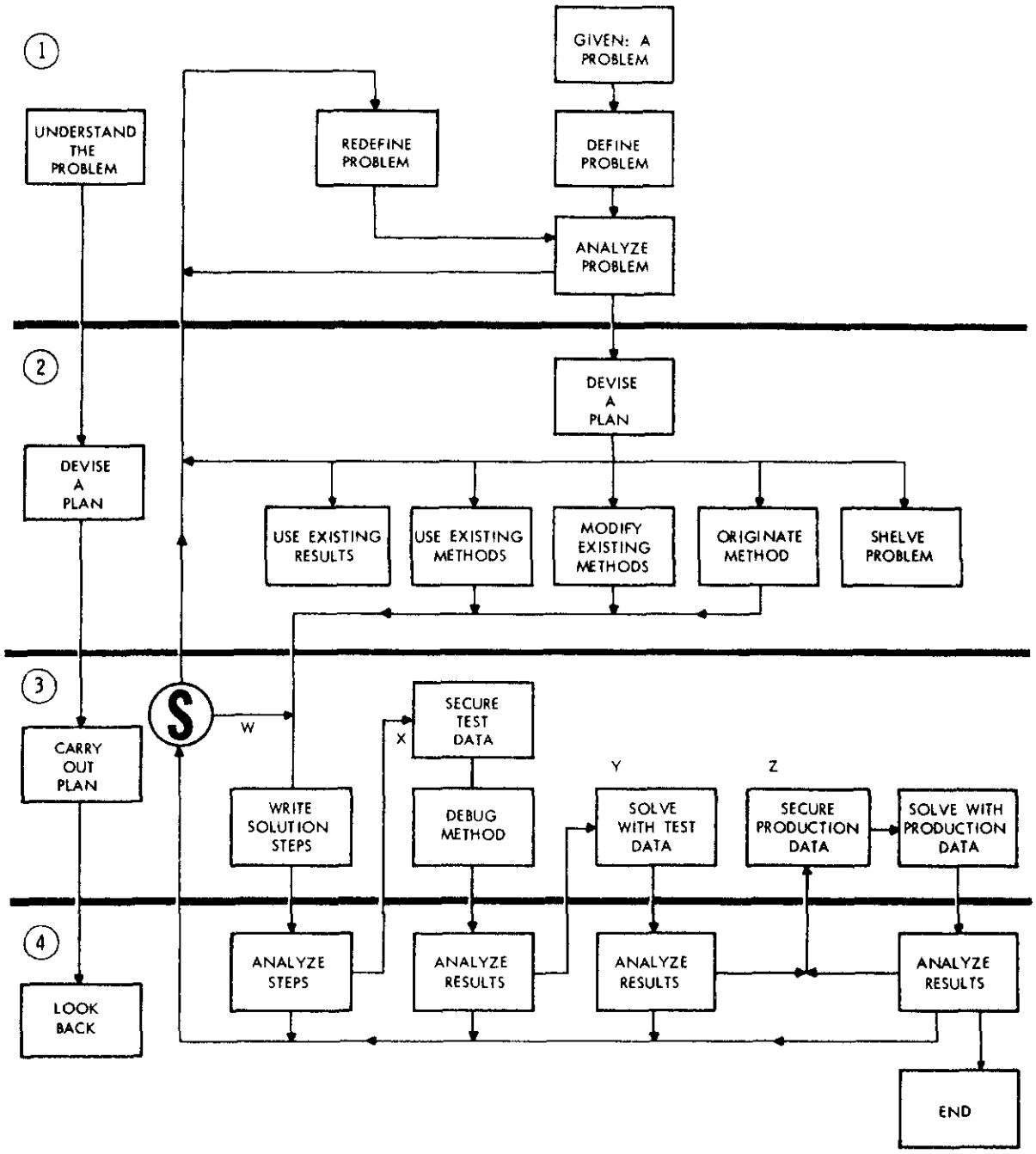


Fig. 1. General Process of Problem-Solving

The introduction of computer assistance into the process of problem solving affects all four steps of Fig. 1. Expanded steps one, two, and four are still representative, but step three must be further expanded in order to represent the carrying out of the plan by the use of a computer. The addition of the remote stations into the process requires further modification of the steps in Fig. 1.

B. LANGUAGES

The present confusion in computer language has been pictured as a modern tower of Babel,[4] with seventy-three different source languages represented as bricks in the tower. Even more computer source languages exist, and each company computing center has its own versions of certain source languages. As is obvious from this multiplicity, no fully satisfactory and universally applicable source language has yet been devised. One of the existing languages may be best for RSC, or a new language may be needed.

Table 2 compares (1) SHARE languages,[5,6] (2) ALGOL 60,[7] (3) BLODI,[8] and (4) MARK 1 Gordon Simulator.[9] SHARE languages, written for use on various IBM computers, include FORTRAN, SAP, SOS, and FAP. FORTRAN (FORmula TRANslation) can express most problems involving numerical computation. It was the first good algorithmic language implemented for widely available machines, and has been in rather general use since it was introduced in 1956 for the IBM 704.

ALGOL 60, also a problem language for expressing processes of computation (algorithms), is a machine-independent source language which has been accepted by groups such as the IRE-PGEC and ACM.[9] At present, ALGOL compilers are available only for the BURROUGHS 220 and the CDC 1604.

"The BLODI source language corresponds closely to an engineer's block diagram of a circuit and is easily learned, even by persons not familiar with computing machines."[7] The MARK I Gordon Simulator source language corresponds to an engineer's flow chart, representing the movement of transactions through a given system. The MARK I is used to study response time and queuing in a given system. The success of these two languages in handling special problems suggests the possible value of a general scientific language with a basic structure corresponding to the engineer's flow chart or block diagram.

TABLE 2
PRESENT SOURCE LANGUAGES FOR COMPUTERS

Language	Type	Introduced by:	Applicable to:	No. of Source Statements	Available Packaged Programs
FORTRAN (FORmula TRANslation)	Program	IBM (1956)	IBM 704 650 700/7000 1620	38 38	SHARE Library
SAP (Share Assembly Program)	Symbolic Machine	United Aircraft (1955)	IBM 704	Machine plus pseudo-	SHARE Library
SOS (SCAT)	Symbolic Machine	IBM and IBM customers (1959)	IBM 709/7090	Machine plus pseudo- plus macro-	SHARE Library
FAP (Fortran Assembly Program)	Symbolic Machine	Western Data Processing Center, UCLA (1959)	IBM 709/7090	Machine plus pseudo- plus macro-	SHARE Library
ALGOL	Program	ACM, GAMM (1958)	Machine-independent. Compiler available for BUR 220, CDC 1604		
SPECIAL COMPILERS: MARK I (Gordon Simulator) (Response time and queuing)	Program (Special)	IBM (1960)	IBM 704/7090	25	
BLODI (Circuit design)	Program	Bell Laboratories (1960)	IBM 704/7090	32	

C. HARDWARE

Table 3 lists significant characteristics of existing and planned hardware which could be used in a remote scientific computing configuration. A more exhaustive listing will be needed in specifying a profitable configuration.

IV. PRIMARY SYSTEM PROBLEMS

Several primary system problems discussed in this section are listed in Table 4, which also (1) indicates who is working on each problem or responsible for its solution, (2) sums up the extent of present activity, and (3) suggests a possible solution for the problem. Two additional RSC system problems, not discussed here, but important in eventual marketing, are: (1) the organization(s) controlling the system or parts (i.e., processor, communication equipment, and terminal equipment) and (2) the type of contract with the user.

A. RELIABILITY, EFFICIENCY, AND ECONOMICS

Reliability, efficiency, and economics should all be considered in relation to each part of the system (i.e., the computer, the communication equipment, etc.) as well as to the system as a whole. Although "reliability" is usually described by both the percentage of time a system is inoperative due to component failure, and by the probability of undetected error which affects output accuracy, the latter meaning of reliability is our concern at the present stage of the RSC project.

Reliability (in this sense) can be approached in the two ways represented in Fig. 2. No scale is given for the ordinate, since accurate

TABLE 3
HARDWARE (EXISTING OR PLANNED) AVAILABLE FOR POSSIBLE RSC CONFIGURATIONS

Equipment	Input/Output Range or Maximum	Remarks	Rental Estimate (Dollars)
Terminal Equipment			
GPD Data Communication Terminal	14.8 char/sec	88 character keyboard 130 char/line printer***	55 - 100
ASD Line Speed Buffer[10]	14.8 char/sec 1200 *** bits/sec up	Transferred to GPD To/from telephone	75
Transmission Equipment			
Dataphone 100	0 - 75 bits/sec		40
Dataphone 200	75 - 1200 bits/sec		
Dataphone 400	0 - 20 char/sec	16 possible characters	
ASD Infrared			
ASD Microwave Radio			
Automatic Answering Unit		Available	
Automatic Dialling Unit***			
Central Equipment (all IBM)			
7281 I Data Communication Channel[11]	2000 bits/sec	32 I/O channels 7 types of sub-channels 709/7090	2, 580*
7284 IV Data Communication Channel	2000 bits/sec	4 channels 7070	RPQ
7750 Data Communication Channel	2000 bits/sec or 200 *** bits/sec	15 channels 100 channels	7000 - 11, 000
7070 Computer			
7090 Computer		Multiprogramming possible	50, 000
729 II Tape Unit	15, 000 - 41, 667 char/sec		700
729 IV Tape Unit	22, 500 - 62, 500 char/sec		900
1620 Computer			1600 - 3200
7281 II Data Communication Channel	160 36-bit - words/sec (parallel)	32 I/O channels 7090 8 types of sub-channels	3580**

* plus $1200 for 7607, 7606, 7100 modifications
** plus $580 for 7606 and 7100 modifications
*** not announced

IBM CONFIDENTIAL

TABLE 4
PRINCIPAL PROBLEMS IN DEVELOPING AN RSC SYSTEM

TABLE : PROBLEMS IN DEVELOPING RSC SYSTEMS

	Problem	Group Responsible	Present Activity	Possible Solution
A	Reliability, Efficiency and Economics	ASD (RSC)	Development of methods of plotting parameters	
B	Human-to-Computer Communication	SHARE Research (Mathematical Science)	SHARE Distribution FORTRAN Development	Adapt FORTRAN for RSC
C	Debugging	Research (Mathematical Science)	Design of special debugging programs	Adapt present methods
D	Verification	ASD (RSC)	Simulation equipment available	Use present methods
E	Communication Between Remote and Central	ASD (Information Transmission)	Low-cost subset Code evaluation	Dial-up telephone and airmail
F	Terminal	ASD (Information Transmission) GPD (Endicott)	Development of Line Speed Buffer Data Communication Terminal	Line Speed Buffer Data Communication Terminal
G	Processor	ASD (RSC)	General specifications	Use IBM 7090
H	Response Time	ASD (RSC) DSD (Scientific TELE-PROCESSING)*	None	Provide 2-3 runs per day per problem
I	Engineer Education	Universities Potential Users IBM Education Department	Not generally available On-the-job training Limited participation	On-the-job training to supplement college training
J	Management Education	IBM Sales IBM Education Department	None for RSC concept	
K	Storage of Source Programs	ASD (RSC)	None	Store at remote station on tape

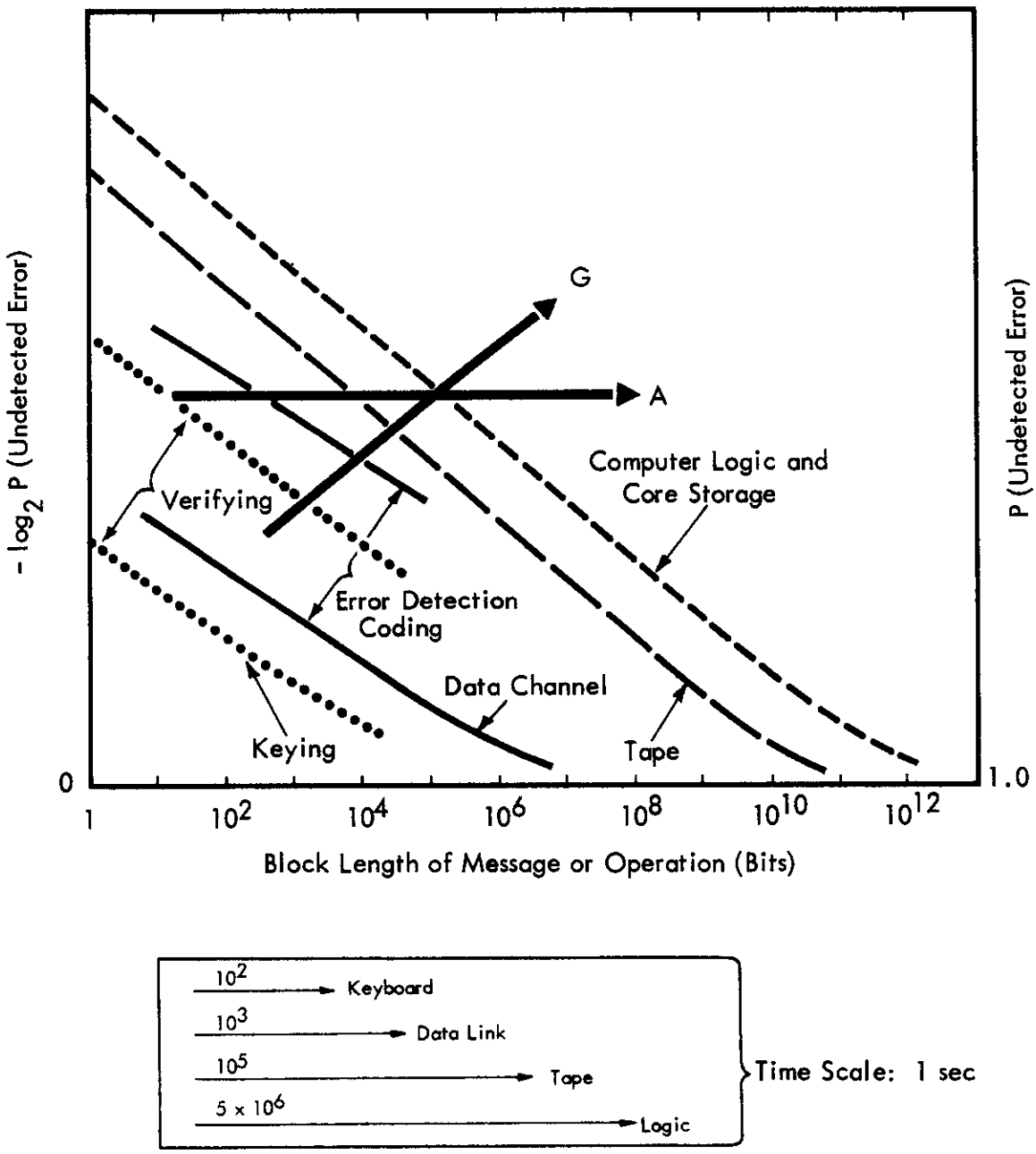


Fig. 2. Probability of Undetected Error vs. Block or Problem Length in Automatic (A) and Gestalt (G) Modes of Operation

values for the curves are yet to be determined. If P is the probability of an undetected error in the character, message, block, or computation, $-\log_2 P$ provides an additive unit for the comparison and addition of errors due to different causes. In Fig. 2, typical curves show the probability of undetected error in keying information (into cards or on tape), in data transmission over voice telephone lines, in information transfer to and from computer tape units, and in computation and information transfer in the computer logic.

The two modes of operation for a computer-communication system are shown in Fig. 2. In the "automatic" mode of operation (line A), the message size, subroutine size, and overall program size are adjusted to minimize the probability of undetected error at each stage. The example shown assumes equal probability of undetected error at each stage, since there is no human review until the end. Line G is an example of the "Gestalt" [14] approach, which in the ultimate limit represents a true real-time system with the operator watching the sample output and making decisions as the computations proceed.

To achieve an acceptable probability of undetected error, line A assumes the availability of an extensive library of subroutines, which are called by short symbols. The structure of the programming system is then adjusted so that, for instance, the keying of 20 bits of instructions could call 10^4 bits of subroutine, leading to 3×10^5 bits being moved in the calculations.

In the Gestalt mode of operation, with a higher ratio of human entry of information during the process, each stage after the input becomes more reliable: the data channel is an order of magnitude more reliable than the human keying, the tape reading an order better than the data channel, and the computer logic an order better than the tape system. In Fig. 2, curve G means that, for example, 2000 bits are keyed by the human operator, 20,000 bits are transferred from tape, and 300,000 bits are transferred in the computation.

The present method of verifying is known to be very accurate compared to unverified input, as the curves in Fig. 2 indicate. The curve for the data channel is higher in reliability than that for human keying, but low compared to that for the computer. By adding redundancy in the form of a cyclic group code to provide error detection with retransmission, the data channel can be made as reliable as the computer logic.

B. HUMAN-TO-COMPUTER COMMUNICATION

Systems study is required to define and resolve problems that exist in human-to-computer communication. Some obvious problems are: (1) What source language should be adapted or developed for remote scientific computing? (2) What logical operations are essential? (3) How should the logical operations be expressed (i.e. in what symbols)?

In considering a possible source language, one must evaluate its efficiency, adaptibility to scientific problems, the time required to teach it, the time required to write new programs, and the availability of packaged programs in the language.

In theory, a source language should be independent of computer and of the native language of the users, but in practice, the source language is generally adapted to fit both. However, the difficulties of converting from one computer to another or from one native language to another should be considered in the study. A universal source language would certainly simplify the training of engineers and scientists in programming. Some efforts to standardize scientific language for computers have been made by committees from such organizations as IRE - PGEC, ACM, NJCC, and GAMM. [9]

It might appear that the source language most similar to human language would be the easiest for an engineer to learn and use. But closer scrutiny shows that the steps in solving most scientific problems are easier to express by flow charts and mathematical formulas than by human language. In fact, a flow chart is usually made before a computer program is written in the source language. The ultimate system may be one capable of translating printed or typed formulas and flow charts directly to computer language. Such a system would minimize the time required for training, programming, and program entry, eliminating the need for (1) translating flow chart and formulas to a source language such as FORTRAN, (2) original keying, and (3) verification (or second keying). However the programmer must have a knowledge of machine limitations. He must recognize that errors in program formulation may occur even in apparently simple symbolic language formulation due to failure to realize the restrictions inherent in the hardware.

Consideration should be given to developing a source language for scientific computing which corresponds closely to the engineer's flow chart or block diagram. Two samples of special programs which use such a source language are the MARK I Gordon Simulator and the BLODI (BLOck DIagram compiler), referred to above. The former is used to study

response time and the effects of queuing upon a given system. The BLODI compiler accepts a source program written in its language and produces a machine program to simulate a circuit.

It is also worth noting that the utility and simplicity of the source language is approximately proportional to the power of the compiler program, as will be discussed in section G. The source language is dependent upon the size of the computer as well as on the ingenuity of those who design the compiler program. As would be expected, experienced programmers can choose among source languages to suit the type of problem, their own know-how, the ease of making program changes, and the stage in which a program exists; while engineers or scientists, less experienced in programming, almost always write, correct and modify their programs in the same source language.

C. DEBUGGING

Another regular system problem is debugging a source program. Information such as that now conveyed by indicator lights to the operator and relayed to the programmer (engineer) verbally or in operator handwriting, must be detected, in an RSC system, by the monitor, or debugging program and be relayed as part of the programmed output. Interpretive routines are available in SOS and are being programmed for FORTRAN 7090 to facilitate source language debugging at execution time.[15, 16] Debugging runs may take priority over certain production runs when the programmer is the engineer himself and is anxious to have the debugging phase completed.

The problem of debugging is illustrated by Figs. 3 and 4, where the probability (P) of an error existing in the program is plotted on a logarithmic scale with a negative sign: higher values on the ordinate represent higher accuracy (i.e., lower probability of error). Curve P_o represents the probability of error as a function of the number of instructions in the program as it is initially written. Satisfactory operation depends on the undetected error remaining below a certain level: P_c represents that level of probability which usually insures correct operation. The scale of the ordinate has been omitted, since we do not have statistically controlled data. Generally a fixed fraction of the programming error will be in the form of inconsistencies which can be detected by the compiler or a monitor program in the first debugging run, moving the curve parallel to P_1. This completes the debugging that the computer can do automatically, though of course consistency

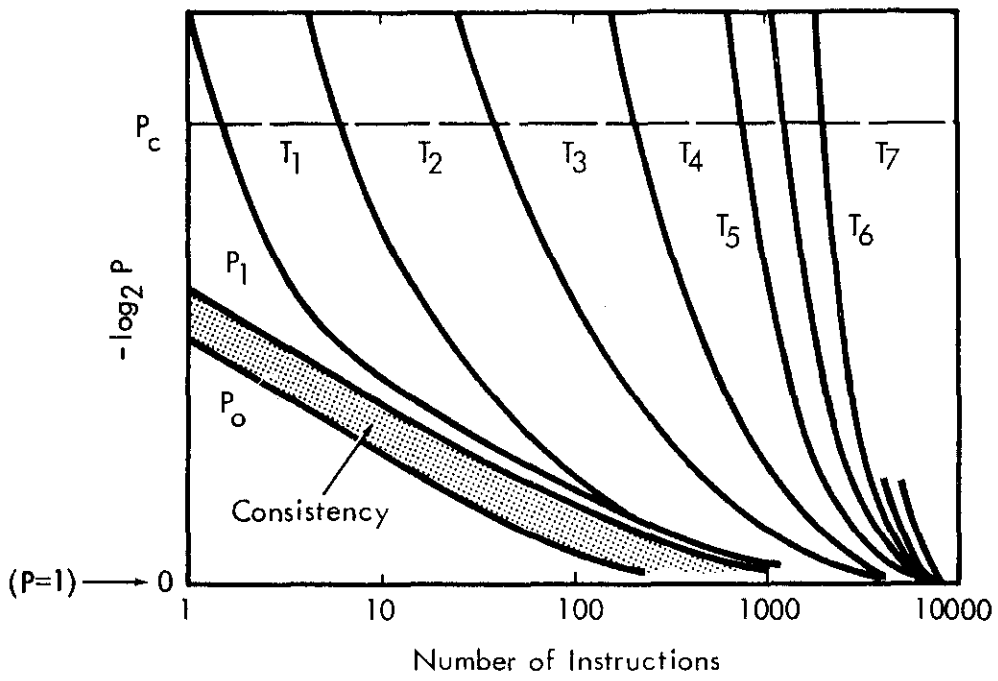


Fig. 3. Debugging a Computer Program by Trial Calculations

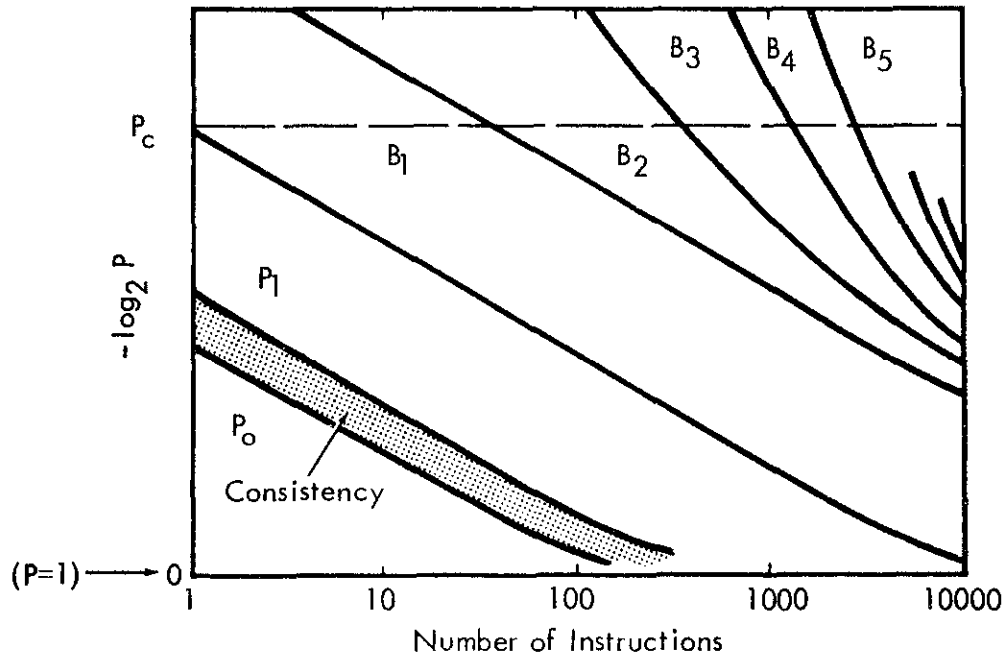


Fig. 4. Debugging a Computer Program by Flow Diagram, with Trial Calculations

should be rechecked after revisions.

The curves T_1, T_2, etc., are drawn,without a numerical scale, to indicate the effect of successive debugging runs using trial calculations. New errors may be introduced in the process of correcting errors previously detected (so that the curves may be double-valued). The shape of these curves is based on the assumption that people generally catch only a few errors per debugging run: this has been partially verified by experience. Short subroutines to be debugged separately are therefore most desirable. The study of the debugging problem, confined to such subroutines, could furnish a reliable numerical scale for the curves. It would then be possible to determine how extensively this form (groups of short subroutines) should be employed in writing programs.

If the debugging runs translate the program into a flow diagram, and in addition print out trial values at critical points in the flow diagram, the first few debugging runs could be expected to detect a fixed fraction of the remaining errors in the program, which would change the curves on the logarithm plot of Fig. 3. After the structure of the flow diagram is corrected, other errors such as incorrect constants could make additional debugging runs less effective for long programs. The hypothetical curves B_1 and B_2 (Fig. 4) show the transition of curves B_3, B_4, and B_5 to the curves T_4, T_5, etc. of Fig. 3.

Without statistical data on debugging, typical numerical values cannot be assigned to the ordinates of these curves, but actual data from operations of the San Jose IBM Computation Laboratory are furnishing some check points.

D. VERIFICATION

When a source program from an engineer's manuscript is put in machine readable form, verification of the transcription is necessary in any system: the addition of a remote terminal, associated with a high-speed computer may both permit improvements and introduce problems in verification. Present methods of verification include comparison between duplicate transcriptions, use of check sums, and visual verification. Verification, as now conceived, checks the clerical process of transcription; editing checks the language and logic of the source program. The study should consider the possibility of reducing verification to a visual check, and using the editing program to detect transcription errors. Thus, redefined to include what is now called editing, verification would

be based upon the internal structural requirements of the source language, and might prove more satisfactory than reliance on duplicate transcription or check sums.

The method of verification employed usually varies with the terminal operator, as well as with the type of information being handled. For example, a regular keypunch operator will invariably verify by "duplicate" keying; while an engineer keypunching his own program is usually satisfied with visual verification. No verification would be needed if the terminal device could transcribe the source program from the engineer's manuscript directly to machine readable language.

E. COMMUNICATION BETWEEN REMOTE AND CENTRAL

Satisfactory communication between the remote station and the central involves several factors which must be studied, including cost of data transmission, cost of terminal equipment, system access and response time, and susceptibility to error. Various means of communication should be considered, including mail or messenger service, which may be feasible in some instances.

Telephone service will more typically provide suitable communication speed and convenience for an RSC system. Certain kinds of applications involving on-line operation may require that a sub-channel at the central be assigned to a leased line. However, many scientific problems do not require leased-line operation, but can be run efficiently in an RSC system with dial-up telephone service and buffered input/output at the terminal station.

The ultimate RSC system may combine both types of telephone transmission, and mail or messenger service as well. Table 5 compares these means of communication on the basis of characteristics significant for remote scientific computing. Further evaluation should compare other factors such as system cost, standby cost, pricing system and response time.

TABLE 5
THREE POSSIBLE REMOTE-TO-CENTRAL COMMUNICATION METHODS FOR RSC

	Dial-Up	Leased Line	Mail
Operation	Buffered input/output	On-line (central buffer)	Buffered input/output
Buffers	Local and central	Central only	Card or tape
Use	Intermittent (for relatively short periods)	3-to 4-hour periods	Intermittent
Line Quality	Telephone	Teletype	None
Central Multiplexing	Telephone Company exchange	IBM Exchange	Manual

F. TERMINAL

Low-cost terminal equipment is needed if the objectives of the RSC system are to be realized. The standby cost must be low for small engineering firms who use the system only intermittently. Terminal equipment chosen or designed must also be suited to the source languages and the general systems specifications. For example, a terminal which could translate a source language of handwritten formulas and flow charts into computer language would certainly be very useful.

G. PROCESSOR

The RSC system designed to serve the classes of users described above must provide a reasonably low cost for intermittent use and a low problem-solving cost. To be profitable, a system meeting these specifications must serve many users, and thus must ultimately employ a large computer at least as powerful as the IBM 7090. The multipurpose monitor and compiler programs, which must be very powerful in themselves, could not be accommodated along with users' programs in a computer much less

powerful than the IBM 7090. A simple comparison of storage capacity makes the problem quite clear: a monitor program occupying 2 to 10% of the 7090 storage (36K 36-bit-words) would require 29 to 144% of the IBM 1620 storage (60K 4-bit-characters). Obviously, a less powerful monitor program would be needed for the smaller computer or the monitor program would have to be called from tape storage between problem runs.

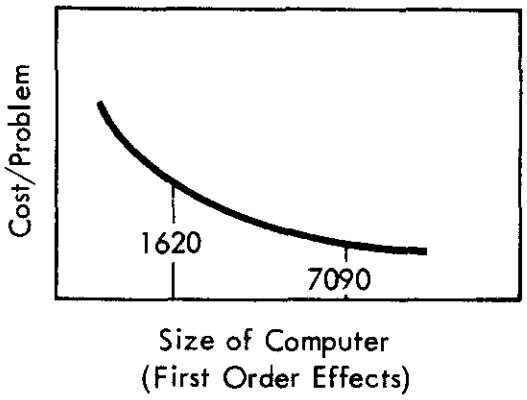
Relative first-and second-order effects upon the cost per problem are illustrated in Fig. 5, which relates computer size and compiler power to this cost. Some related graphs, having numerical coordinates, have been prepared in another study,[17] which also evaluates computers in terms of relative utility per dollar ("additions" performed per dollar). A summary of the relative utility of some IBM processors is given in Table 6, derived from information presented in that study.

TABLE 6
RELATIVE UTILITY OF SEVERAL IBM PROCESSORS

IBM Processor	Relative Utility*
Stretch	500
7090	20
709	4.8
704 (1958)	4.3
704 (1956)	1.1
1620	0.82
701	0.36
650	0.17

* "Additions" per dollar.

An RSC system using a 7090 computer might affect the sale of small systems such as the 1620, and this possibility should be considered. However, an overall increase in computer use can be expected as a by-product of RSC systems, and the demand for small computers could actually increase.



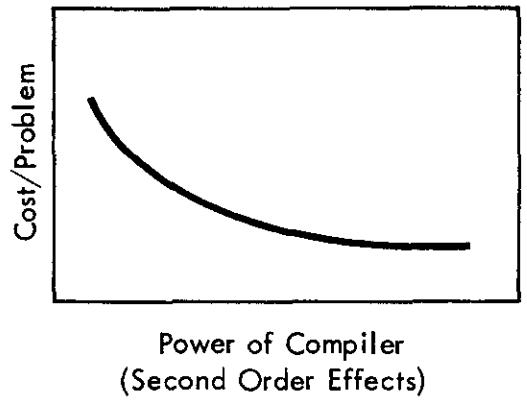


Fig. 5. Cost per Problem, Related to Computer Size and Compiler Power

During the marketing stage of an RSC system, the central computer could be used in other applications, with multiple use of the computer decreasing as the RSC load increased to full capacity. With a central exchange compatible to IBM 729 tape systems, any of several processors could be selected, or used in turn as the RSC load increased. Designing the RSC system should involve, wherever possible, matching existing computers and computer systems with the needs of RSC.

H. RESPONSE TIME

Optimum response time, another important systems consideration, involves several factors: relative cost of various response times, type of run (i.e., compilation, debugging, or production), problem application, and user's requirements. These must all be considered in defining satisfactory response time for a given application.

One psychological difficulty, the engineer's feeling of loss of control once his problem is committed to the computing center, can be minimized by rapid response from the computer - - reported by telephone for instance, rather than by messenger. After either selecting an existing computer program to solve a particular problem, or preparing a new program (which may take from several hours to several months), the engineer sends the program to a computer for a trial or debugging run. The results and error statements are returned to the engineer, who examines them, makes corrections and returns his program to the computer. Five or more debugging runs may be needed to completely debug the program, followed by one or more production runs to solve the problem.

A day or more may elapse between runs when the engineer and the central computer are at some distance, and an appreciable time delay may occur even when they are in the same building. But with a properly designed RSC system, three or more runs per day could be made and reported regardless of the distances involved. Two runs per day are considered minimum for effective remote scientific computing service.[18] Thus two questions regarding response requirements have been raised and should be answered: (1) How quickly should the engineer receive the results of a run after he has entered the program into the system? and (2) How many runs per problem per day must the system be capable of doing?

I. ENGINEER EDUCATION

To make good use of RSC service, an engineer must understand how a computing system works, know how to communicate with the system, and have a knowledge of computer applications. Education for this is not generally available in college courses, although 71 colleges and universities are listed[19] as offering training in electronic computers. Recent reports indicate a still unmet need for courses giving (1) an appreciation of the computer as an aid to routine mental effort, (2) a theory of computers (3) a theory of programming, and (4) a theory of applications.[20] The need for such a curriculum, as described in "The Computer-Related Sciences at a University in the Year 1975,"[21] must be impressed upon college and university administrators.

H. L. Colman, Chairman of SHAREmanship subcommittee on Installation Training Programs, states that "students are rarely given a programming course, but rather a course in how to code in some language," and that the only recourse for businesses and industries is on-the-job training.[22]

At present, university computation centers play a larger role than does college curriculum in the computer education of future business administrators, engineers, and scientists.

J. MANAGEMENT EDUCATION

Business, industry, research, and educational facilities could obtain much greater benefits than they do from existing data-processing systems. Though scientists and engineers have recognized computers as powerful, valuable tools, management in general in business and industry has been slow to react to the potential that computers offer.[23,24] The use of computers has now penetrated into large and medium size businesses and industries in the United States, but remote scientific computing systems will extend the service of large high-speed computers to small businesses and industries, and augment the established use of computers in large and middle size businesses and industries by providing internal RSC systems.

Successful marketing of remote scientific computing systems will depend on the education of management, as well as of the engineers and scientists, in the value of the computer as (1) an aid to routine

mental effort, and (2) a tool to provide new capacity, and to extend and refine control of the business or industrial operations. Management education must include the clear understanding of the limits of legal responsibility for decisions based on computer solutions to business problems.[25]

K. STORAGE OF SOURCE PROGRAMS

Study should be made to determine the best storage location for source programs not in regular use by two or more remote stations. If a source program is stored at the remote station, the data transmission requirements are greater since repeated entry of the source program is required during the debugging stage. Storage of all source programs at the central would create extra housekeeping problems there.

V. POSSIBLE SYSTEM CONFIGURATIONS

A general configuration for a remote scientific computing system and the operation of such a system can now be proposed and compared with other possible configurations, including one considered by IBM France.

A. GENERAL CONFIGURATION

No specific configuration can be presupposed at this point, but a general configuration (Fig. 6) and operation steps, suggested here, can help define remote scientific computing.

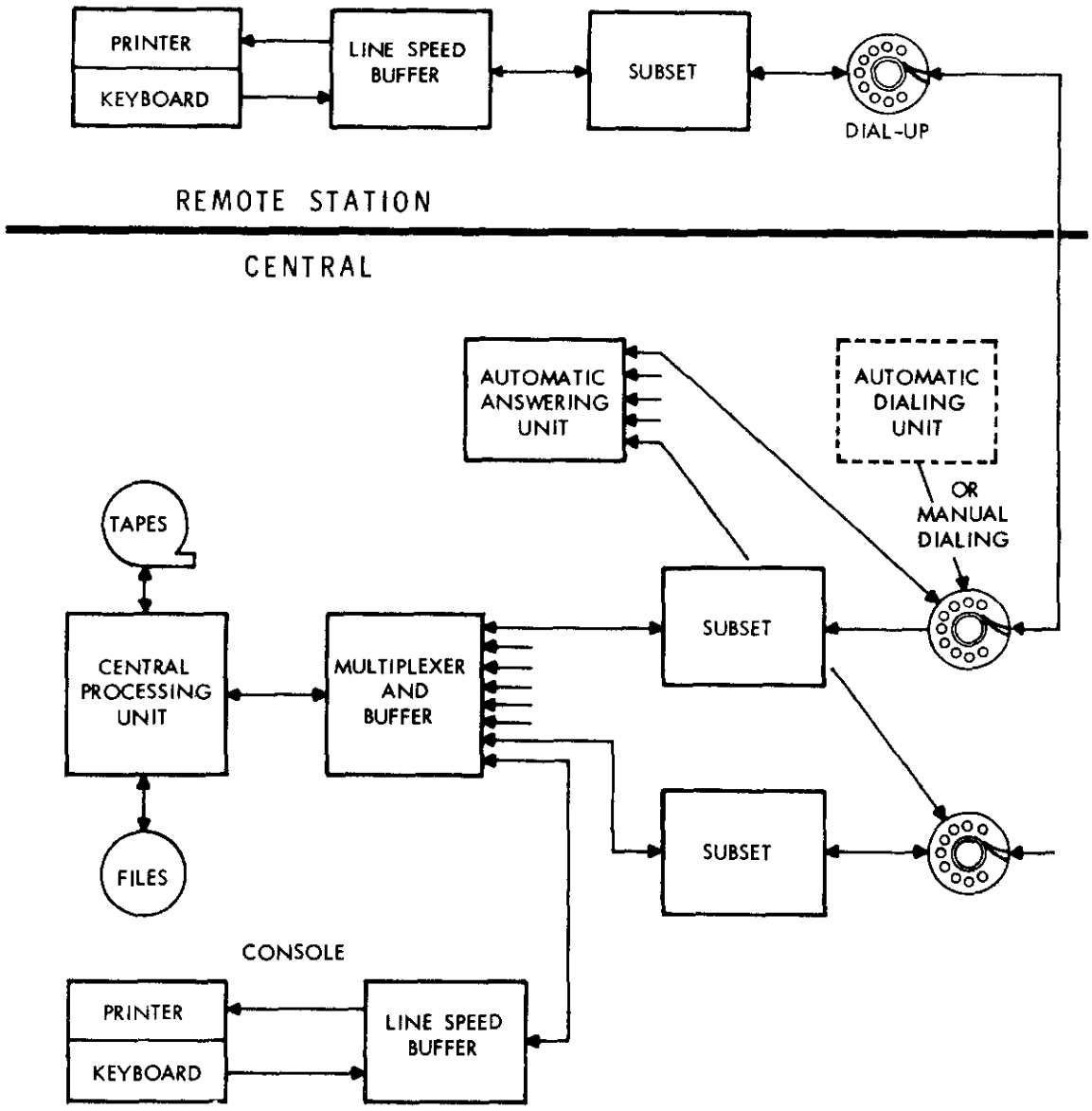


Fig. 6. Possible RSC System Configuration

The remote station indicated contains a terminal (printer-keyboard), a buffer, and a means for communication with the central, (possibly a data subset and dial telephone). The central contains a central processing unit, a multiplexer, and a number of sub-channels for communicating with remote stations.

The operation of the system is illustrated in Fig. 7 which depicts the following steps:

1. Programming. A typical scientific problem may require several weeks or several months to program.

2. Program Entry. The source program is entered into the system via the typewriter. As each character is entered, it is written on the magnetic tape of the line speed buffer. Then the tape is advanced one character position, and the recorded character is read to the printer.

3. Transmission to Central. The line speed buffer tape is reversed and its content (e.g., source program and/or data) transmitted over a dial-up telephone line at 1200 bits per second to the central buffer. (Higher rates are also possible).

4. Processing. The monitor program inspects the source program, selects required programs from its library, and determines the optimum scheduling of the programs to be run. The source program is then run, and the results of the computation transferred to the central buffer. (These results could, of course, contain error statements or, in debugging, selected memory dumpout.) Programs may be run sequentially to avoid multiprogramming or may be multiprogrammed to use the central processor more efficiently.

5. Transmission to Remote Station. The remote station is dialed manually (or automatically) and the results are transmitted at 1200 bits per second, and recorded on the remote line speed buffer.

6. Printing of Results. The line speed buffer tape is reversed and read to the printer at typewriter speed (e.g., 14.8 characters per second), producing a printed copy of the results.

7. Analysis of Results. For a production run, the cycle is complete when the results are received and printed. However, new data may be entered, and the cycle resumed at step two. For a debugging run, the engineer would correct the program and the cycle would be resumed at step two.

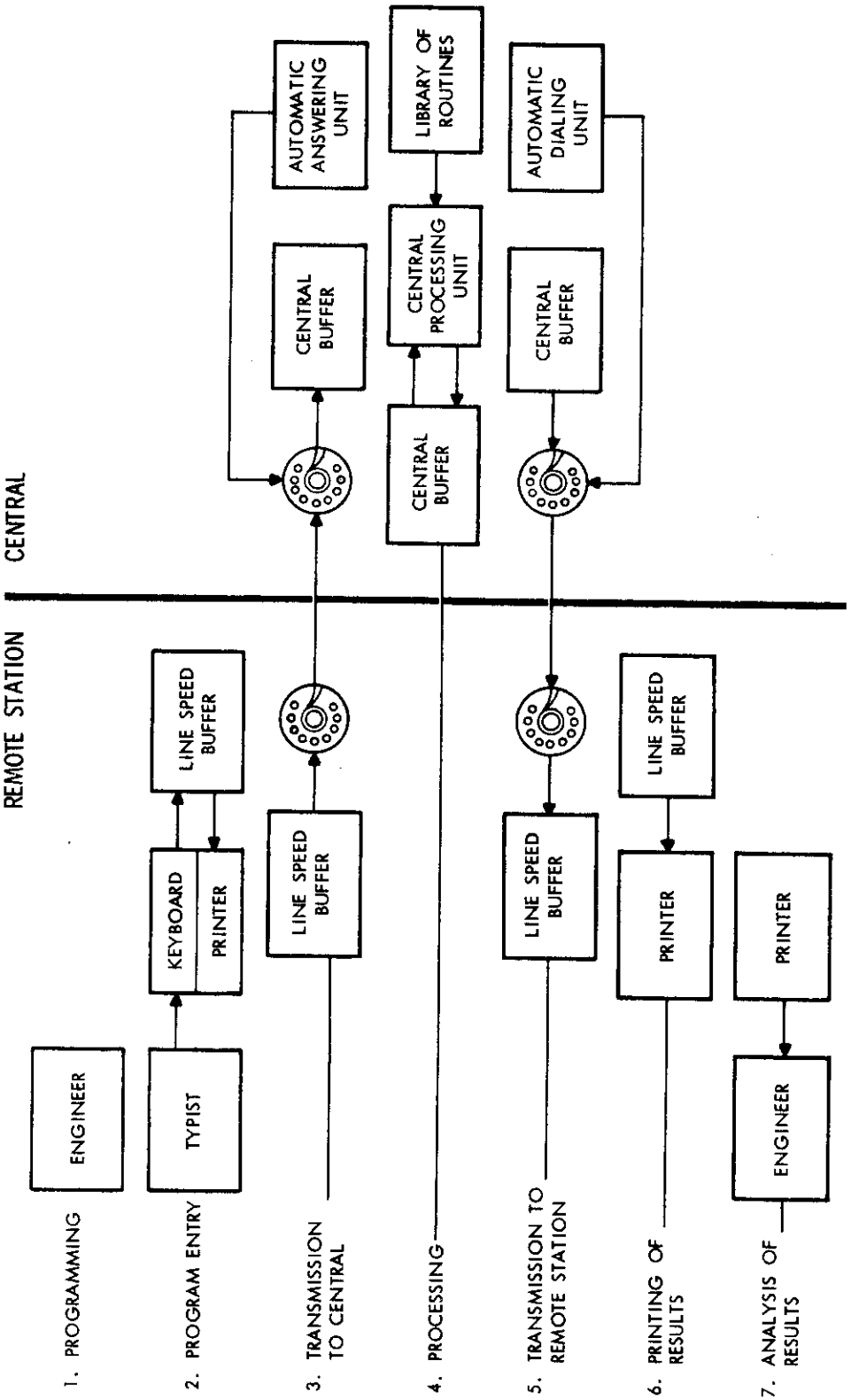


Fig. 7. Operation of Possible RSC System

Figure 8, an expansion of Fig. 6, shows some possible combinations of units for a remote scientific computing system. While this kind of configuration might solve certain scientific problems for a certain number of remote stations, it is not being offered as an ultimate or even an intermediate solution for the remote scientific computing problem. The role of the multiplexer in Fig. 6 is performed in Fig. 8 by the set of units within the dotted area. The dotted boxes represent components not now available, but which could be designed using existing techniques.

B. ALTERNATE CONFIGURATION - IBM FRANCE

R. Corby of IBM France has expressed interest in developing an RSC configuration similar to that of Fig. 6, except that the multiplexer would be replaced by a telephone channel control unit patterned after their Telegraph Channel Control Unit (TCCU). In this suggested configuration, the TCCU links thirty teletype stations to the central processing unit, and plays the part of a 729 II or 729 IV tape unit in relation to the computer.

C. OTHER CONFIGURATIONS

Other configurations which could prove useful may omit the local buffer and include leased lines, telegraph lines, and mail service; or they may involve a hierarchy of computers connected by wideband communication channels.

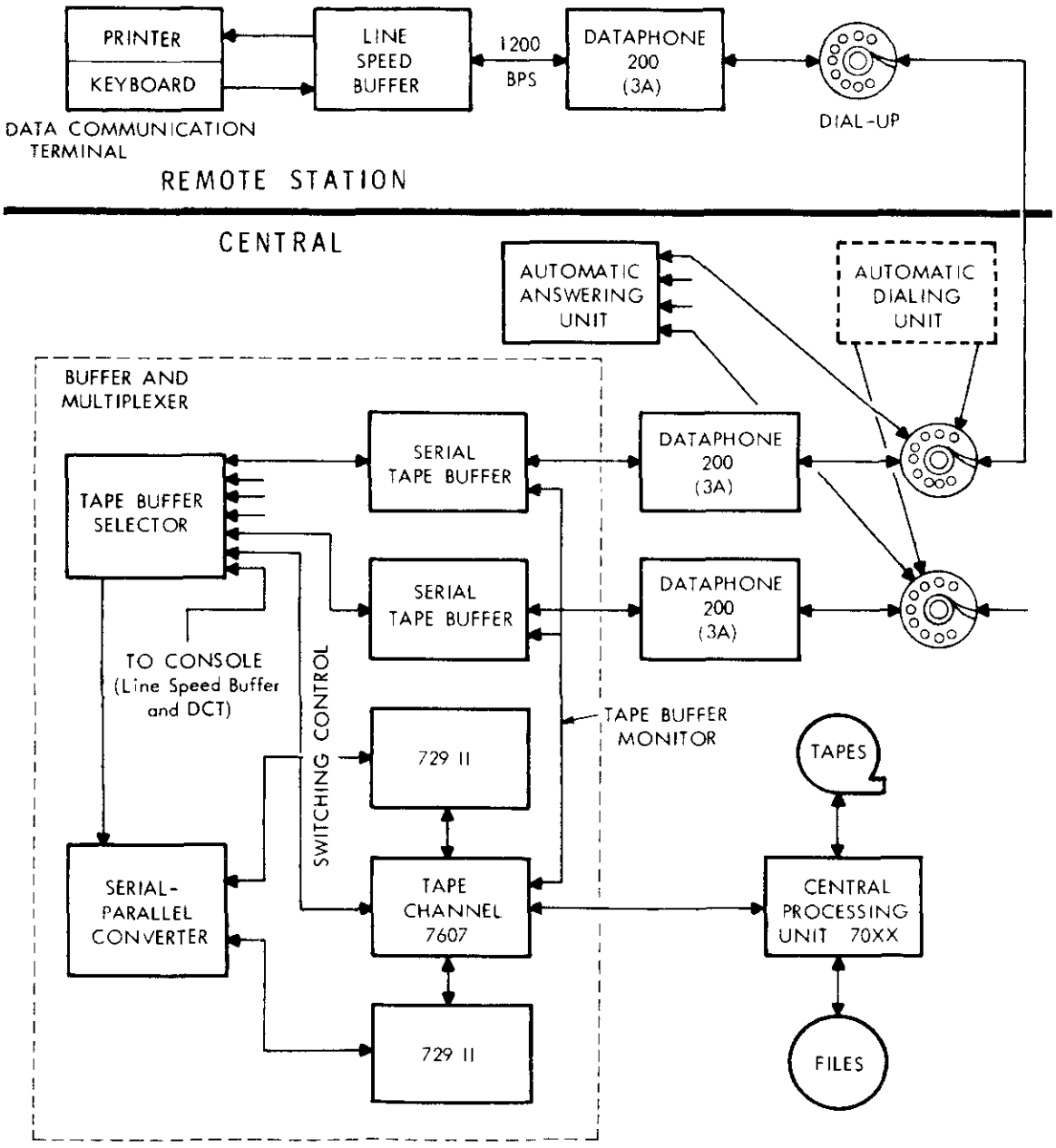


Fig. 8. Possible Combination of Units for an RSC Configuration

VI. RELATED ACTIVITIES

Several IBM activities, including the following, are related to the objectives of the remote scientific computing project, and are available sources of information for this study:

1. Scientific TELE-PROCESSING*

A new group formed in San Jose under Dr. E. C. Smith of DSD, TELE-PROCESSING, will conduct a systems study to help evaluate proposed additions to our product line. As a part of their study, to gain operating experience, this group will design and implement a system involving the 7090, with several remote terminals on-line and a multiprogramming monitor routine.

2. IBM Systems Research Institute

G. M. Weinberg of the IBM Systems Research Institute (a graduate education and research center for systems engineers) is interested in remote scientific computing and is encouraging the writing of term papers on this subject to fulfill course requirements. Copies of such reports will be forwarded to this project: the first such paper [18] has been received.

3. San Jose Development Laboratory

Dr. E. F. Lindstrom, Manager of the Engineering and Scientific Computation Laboratory, is promoting open shop use of high-speed computers by engineers in the San Jose Development Laboratory. Results of this activity will provide practical information for the RSC project regarding kinds of problems submitted and user requirements in such a situation.

4. Yorktown Heights Research Laboratory

Lois Haibt, of the Yorktown Heights IBM Research Laboratory, is designing a routine for the 7090 which produces a flow chart of a source program. [26] A routine of this kind may be useful for the debugging of source routines by the RSC system.

*Trademark

5. GPD Processing Systems, San Jose

John F. Holmes, Resident Manager of Processing Systems in the San Jose Development Laboratory, who is responsible for the engineering and marketing of the 1620 scientific data processing systems, is interested in the development of a remote terminal to be associated with a 1620 data processing system for scientific applications.

6. ASDD, Mohansic

R. E. Nienberg, of ASDD, Mohansic is interested in multiprogramming, suited to the purposes of RSC, in a modified 7090.

7. Federal Systems Division

FSD has had a considerable amount of experience in remote operation and data transmission in relation to military applications.[27]

8. DSD, PDL, Poughkeepsie

The work in multiprogram scheduling[28] and simultaneous operation on programs,[29] being carried on by E. F. Codd in Poughkeepsie, may have results applicable to RSC.

9. University Computing Centers

In the past ten years the number of university computing centers has increased from about a dozen to more than a hundred,[30, 31] and several of them are actively interested in remote scientific computing: the Massachusetts Institute of Technology, Stanford University, the University of California, the Carnegie Institute of Technology, and the University of California at Los Angeles, among others.[19]

The Bendix G-20 system, installed in the Carnegie Institute of Technology Computation Center in May 1961, provides for remote consoles from which students and faculty may request execution of programs recorded permanently at the central computer.[32]

10. Canadian Universities

In 1955 and 1956, one of the first remote computing projects involved the Ferranti computer at the University of Toronto, and communication over teletype lines.[33, 34] The Canadian National Railroad made its teletype network available during the night to universities for sending computer programs (in teletype punched paper tape format) and input data to the computation center at the University of Toronto. The machine supervisor there would check the program, run it on the Ferranti Mark I, and send the results back to the originator via teletype. In 1956 fifteen people at the University of Saskatchewan, who had never seen the computer at Toronto, ran successful computer programs via the teletype link. Knowledge of the details of these experiments may be useful in RSC analysis even though the speeds and other conditions are not comparable to those now contemplated.

VII. RECOMMENDATIONS

On the basis of the preceding analysis, the problem of developing an RSC system can be summarized as in Fig. 9. This arrangement lists the principal aspects of the problem as analyzed here; then indicates which of four techniques are applicable to each step in RSC development -- theoretical analysis, simulation tests, experimental systems, or commercial evaluation, and applies each of these methods, in logical order, to the sub-problems named. The ten consecutive steps group different parts of the problem for the various IBM departments involved, and arrows show how the solution of one problem leads to the solution of others. These relationships are described briefly below.

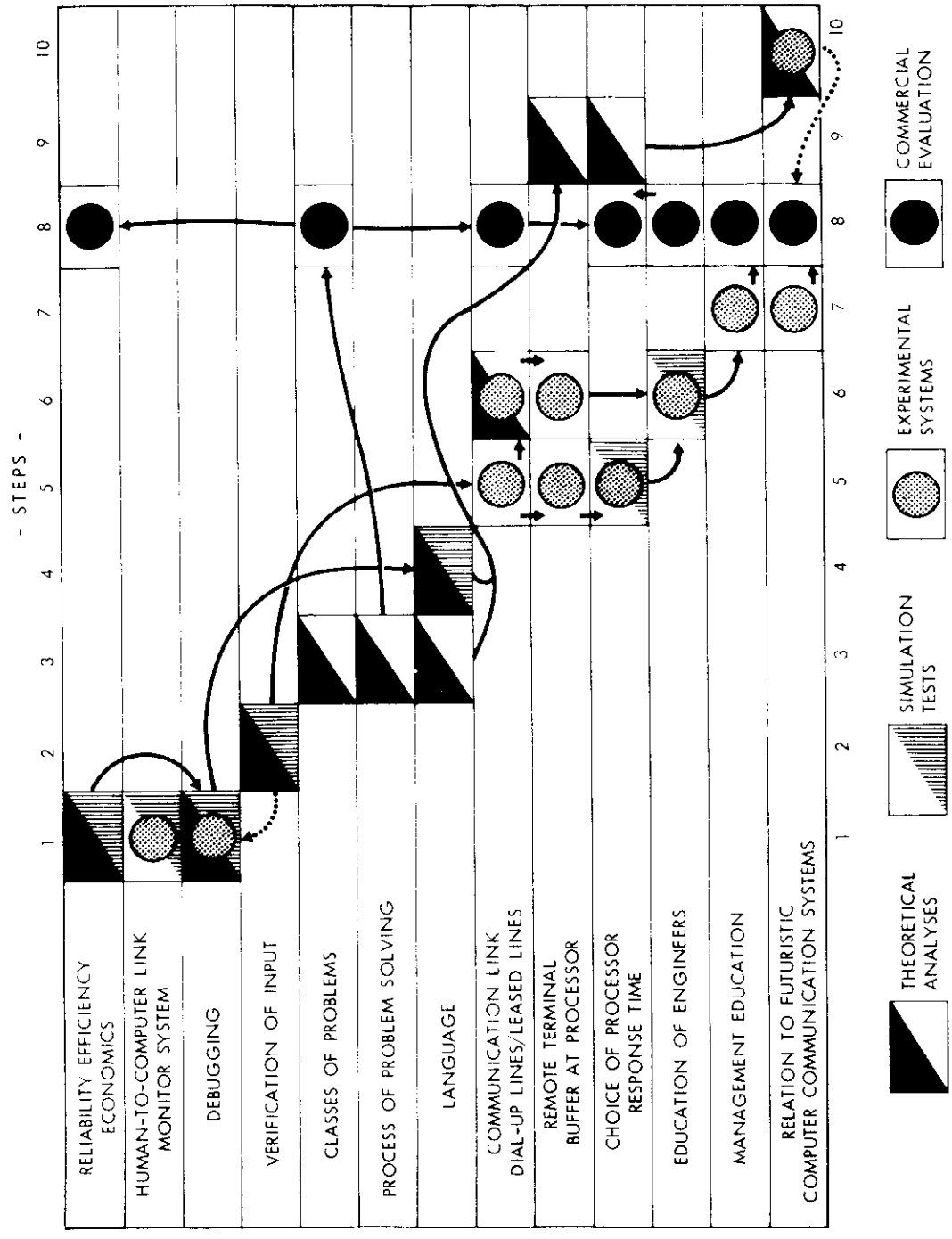


Fig. 9. RSC System Development: Relationships Between Different Steps of the Problems

1. Reliability, Human-to-Computer Link, and Debugging

This involves primarily theoretical analysis and simulation of examples of the ultimate system on an existing computer. Some theoretical analyses of the debugging process and of the probability of undetected errors in a computer system are required. A monitor program must be written to handle (a) debugging, and (b) production, with debugging having priority, and other requests left in queue. Particular techniques useful in debugging can be separated from this group: for example, computer-generated flow diagrams of programs.

2. Verification

This study must determine the feasibility of a relaxed verifying system (such as the proofreading of typed programs) instead of double keying as is now done with punched cards. This feasibility study can be made separately on an IBM 870 document-preparing machine, or on a line speed buffer which more accurately represents a hardware configuration being considered for the remote terminals.

3. Study of Types of Problems

This study would lead into the problem of computer language. The thorough analyses of the types of problems would permit a more detailed commercial analysis.

4. Language

The problem here is to determine the RSC requirements on a language, and to see how the choice of language affects the system. It may be necessary to consider a universal language suitable for both automatic and Gestalt operation modes.

5. Experimental Systems

This would be an experimental system using the latest practical techniques including such equipment as the line speed buffer. As Fig. 9 indicates, the planning of an experimental system would depend on previous reliability and debugging studies (step 1). The remote terminal design would depend on the verification study (step 2). The experimental system study should have available the results of a theoretical system analysis (step 9) to help in making such decisions as the choice of a processor.

It would be desirable to test two experimental systems: one for a large computer such as the 7090, and one for a small scientific computer such as the 1620. The smaller system could be a simplified version without data links, using messengers to carry tapes. These two approaches could give IBM the necessary experience and equipment to plan smooth transitions from present procedures to remote scientific computing systems.

6. Preliminary Demonstration System

The object of this system is to provide a terminal and computer adapter to fit any present computer using 729 tape units and to provide preliminary systems for engineering and management education. This project could be handled by a different laboratory from that running the more general experimental system.

Such a demonstration system would permit an earlier general commercial analysis than would otherwise be possible, providing a preview of the way actual demonstrations can help educate management and engineering personnel in regard to remote scientific computing. Of course caution must be observed to make sure that an uneconomical demonstration system is not considered as a production system.

7. Preliminary System for Education

The development of the experimental system (step 5) could provide the backbone for extensive educational programs to build market interest in remote computing systems. Preliminary educational work could be initiated by a simpler demonstration system (step 6).

8. Commercial Evaluation

The commercial evaluation can be made on a sound basis after some preliminary results have been obtained in the problem solving analysis, and the probable effectiveness of engineering and management education programs has been estimated. A firm commercial evaluation can best be obtained by a field test on a customer's premises, providing actual data for some trial values on the economics of computer usage in remote scientific computing. In Fig. 9, both dial-up and leased-line sections are shown, since both may contribute substantially to the revenue in such systems. When the commercial analysis is completed, the choice of a processor can be made on a sound basis.

9. Theoretical Model of Usage

A theoretical analysis of the load and queuing would help establish possible values for response times with different processors and with different problem distributions. After the initial theoretical analysis, the next step would probably be a simulation study.

10. Relation to Futuristic Systems

Some preliminary studies of the relationship of remote scientific computing to other systems would also help the commercial evaluation. If, for instance, remote scientific computing is expected to merge with multiple business systems in ten years, this estimate might weight some present choices in design. It would be useful to know if some problems of remote scientific computing lie in the chain of steps needed for other systems such as Lasswell's "Social Planetaria"[35] or real-time language translation on international telephone circuits.

The above problem areas have been grouped to facilitate the discussion of the timing of the development of RSC through its various stages, and to simplify the assignment of responsibility for various parts of the problem.

VIII. CONCURRENT SYSTEM STUDY AND MARKETING OF SMALL SCIENTIFIC COMPUTERS

In view of the present marketing problems in RSC, "manual" systems seem feasible for present installations for limited cases. Such installations could then be studied as part of a thorough systems study of the complete RSC area. Several phases are suggested which follow an increasingly complex system configuration and would run consecutively with the systems studies.

The five phases (or configurations) suggested are depicted by Figs. 10, 11, 12, 6, and 13 respectively. The first three are generalized configurations of three specific equipment organizations recommended by the Processing Systems Development Laboratory, San Jose, for their 1620 low-cost scientific system, which has as input/output equipment a typewriter and a magnetic tape buffer with portable tape cartridges.

In the first phase (Fig. 10), Processing Systems envisions the preparation of input data at several in-plant remote locations, with the cartridges hand-carried to the 1620. The 1620 operator would, in this case, provide priority control and preparation for processing by setting up a blank cartridge for the results and entering the required assembly routine or previously developed program. After processing, the results, in a magnetic tape cartridge, would be returned for printing at the remote location.

In phase two (Fig. 11), the hand-carry is replaced by an in-plant wire system utilizing a line-speed buffer. A single transceiver, convenient to a group of magnetic tape typewriters, sends data to and receives data from a similar unit in the 1620 room. The operator continues the priority and monitoring control in this phase.

In phase three (Fig. 12), the knowledge gained from the previous phases makes it possible to prepare priority and monitoring programs to control interrupt calls from directly connected magnetic tape typewriters which receive data when ready to process, and return the results to a blank cartridge at the remote station. This phase uses interrupt and multiplexing techniques.

The first three phases recommended here can be, in effect, the experimental systems approach recommended in section VII for solving the remote scientific computing problem. By marketing these first three phases and making the system study concurrently, IBM can benefit, and so can the users of the system, who will be paying for our experimental system while solving their own problems by remote scientific computing. During this concurrent operation, several problem areas (see Fig. 9) can be examined closely, and solutions formulated for such problems as education of engineers and management, communication link, remote vs. central buffering, and response time. The human-to-computer link and debugging problems can also be considered during these first three phases.

Phase four (see Fig. 6), would begin when the number of remote stations in a given area warranted consolidation into a single system

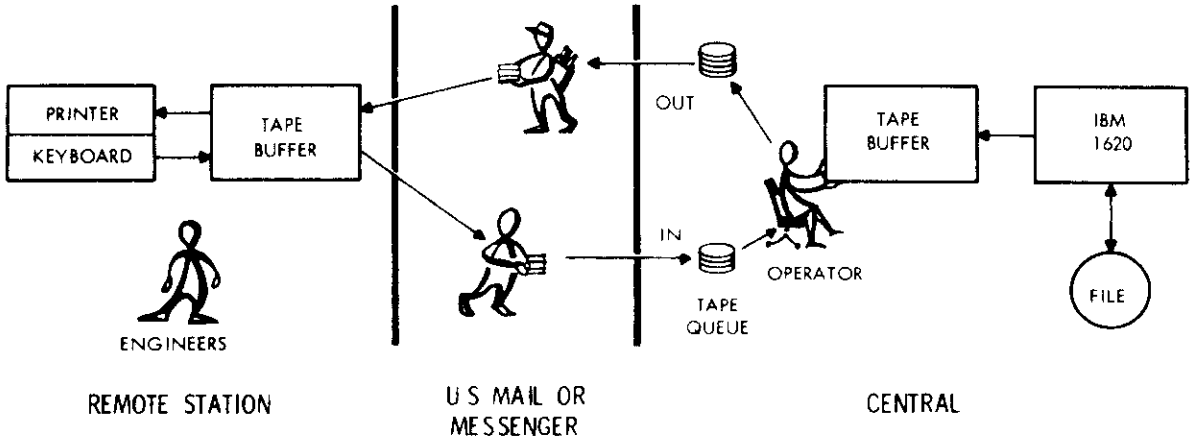


Fig. 10. RSC Phase One

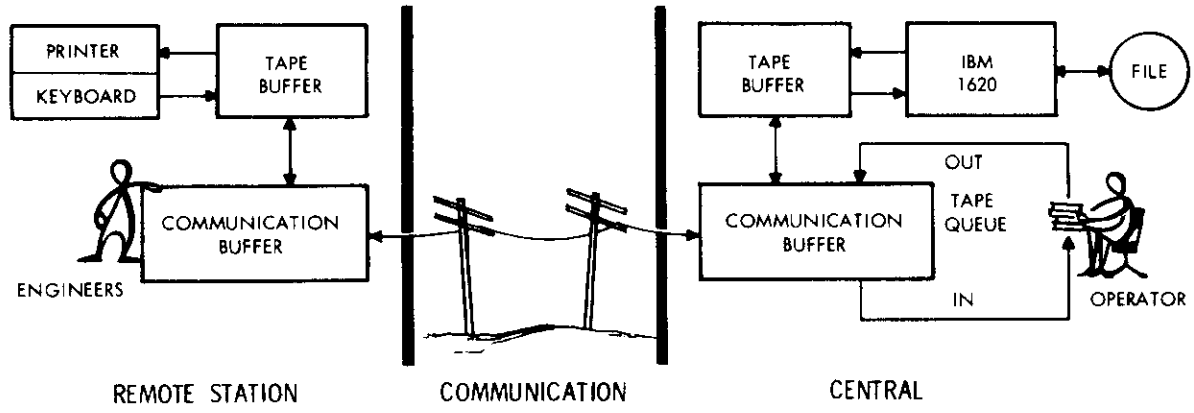


Fig. 11. RSC Phase Two

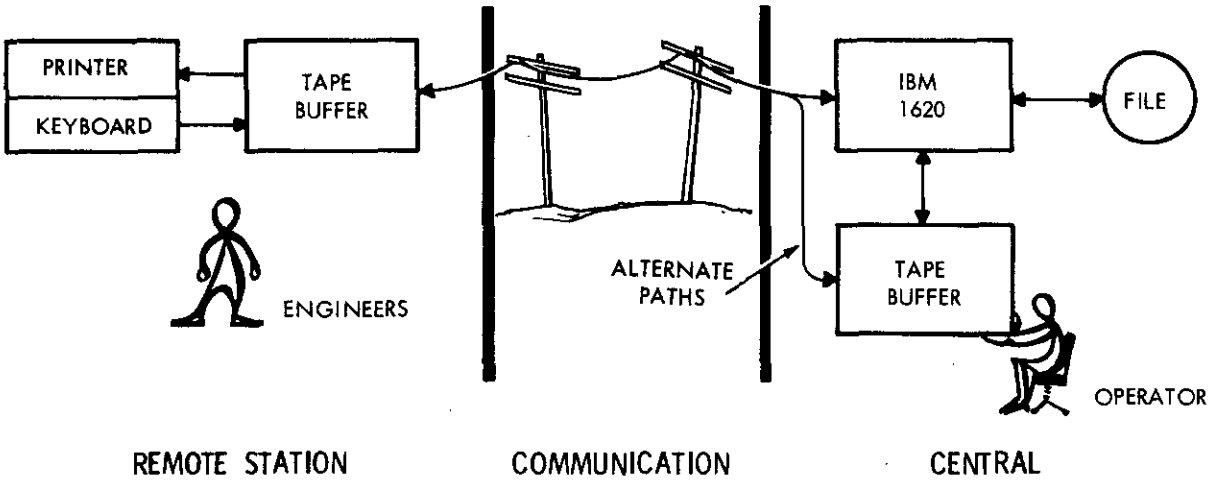


Fig. 12. RSC Phase Three

NOTE: RSC Phase Four (see Fig. 6)

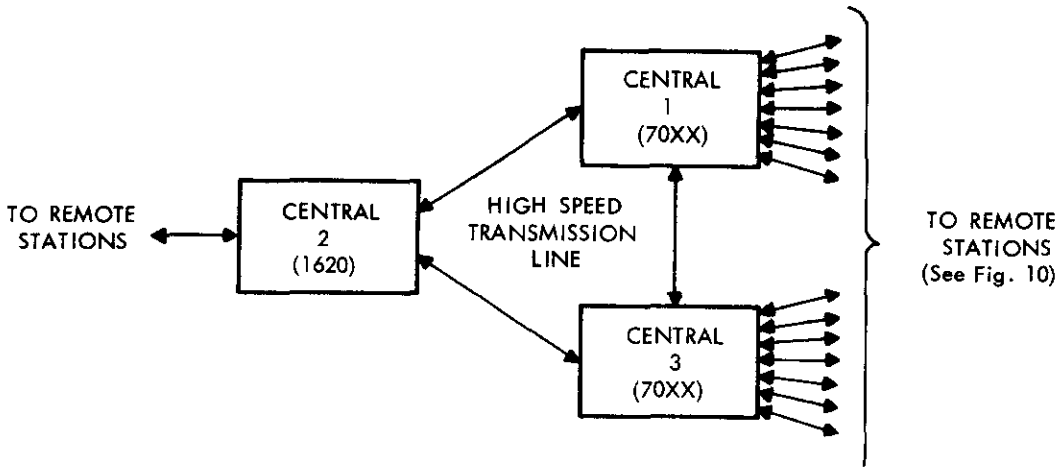


Fig. 13. RSC Phase Five

served by one large computer, which would then take over the bulk of the monitoring, priority assignment, and debugging. Considerable study and experience would, of course, precede this phase. But if the marketing of the systems of phases one to three ran concurrently with the problem study recommended in section VII, the study and actual field experience would complement one another. In phase five (see Fig. 13), a hierarchy of central processing computers of various size, in various locations, is linked by high-speed communication equipment.

ACKNOWLEDGMENT

By their valuable counselling, these men have contributed to the work reported here, and their help is gratefully acknowledged:

R. D. Anderson, then Project Coordinator, ASD Commercial, San Jose, (now Manager, Market Planning, Communications and Special Systems Engineering, GPD, San Jose)

R. M. Bennett, Manager, Technology Development, ASD, San Jose

R. Corby, Assistant to Manager, Product Development, IBM France

J. F. Holmes, Resident Manager, Processing Systems, Development Laboratory, GPD, San Jose

E. Hopner, then Manager, Data Transmission, ASD, San Jose, (now Manager, ASD Computer Communications Systems, Mohansic)

Dr. S. L. Jamison, Manager, MBS Mathematics and Programming, ASD, San Jose

Dr. H. G. Kolsky, Manager, Systems Science, Research Laboratory, San Jose

Dr. E. E. Lindstrom, Manager, Engineering and Scientific Computation Laboratory, GPD, San Jose

J. G. McPherson, Vice-President and Director of IBM Systems Research Institute, New York

C. V. Siebs, Senior Industry Analyst, ASD Commercial, San Jose

Dr. R. M. Simons, Manager, Computation Laboratory, ASD, San Jose

Dr. E. G. Smith, Jr., Manager, DSD Scientific TELE-PROCESSING* Systems Applications, San Jose

G. M. Weinberg, IBM Systems Research Institute, New York

* Trademark

APPENDIX: COMPARISON OF REMOTE SCIENTIFIC COMPUTING AND MULTIPLE BUSINESS SYSTEMS

The obvious similarities in RSC and MBS (Multiple Business Systems) raise a natural question about their significant differences. The two kinds of systems are compared in the following tabulation, which lists the characteristics of each predominant at the date of this report. Where similarities occur, overlapping characteristics are indicated.

APPENDIX:
COMPARISON OF REMOTE SCIENTIFIC COMPUTING AND MULTIPLE BUSINESS SYSTEMS

System Characteristics	Remote Scientific Computing	Multiple Business Systems (as of October 1961)
1. Reliability	Important, but results from single run usually backed up by similar runs and experiments.	Critical: undetected errors have immediate consequences.
2. Efficiency and economics	Large computer, serving many terminals, which help distribute cost.	
3. Human-to-computer communication	Input: FORTRAN or similar types of statements.	Input: predominantly prescribed format; some source language input.
4. Debugging	Highest priority: special programs and procedures (see item 22, below).	Low priority: handled during periods of low activity.
5. Verification	Visual verification and computer editing based on internal structural requirements of source language.	Similar to RSC, but more critical because of immediate results. Check sums also used.
6. Remote-to-central communication	Dial-up: practical for intermittent use anticipated. Telephone bandwidth or greater (i.e., 1200 bits/sec or more). Long and short distances.	Leased line (questionable efficiency). Telegraph bandwidth sufficient (i.e., 200 bits/sec). Short distances.
7. Terminal	Low cost: typewriter and tape buffer.	Low cost: input/output typewriter plus typewriter printer.
8. Type of central processing unit	IBM 7090 (?)	IBM 1410 X (?)
9. Response time	Delayed processing (e.g., 2 or 3 hour response time) probably acceptable.	Varying, but much more rapid: 0.5 sec for line of data (checked for field length and class of characters (i.e., alphabetic or numeric) 5.0 sec for inquiries 10.0 minute response considered delayed processing.
10. Programmer education (see item 26, below)	College education plus on-the-job training.	IBM training.
11. Education of prospective user	Must sell use of computers for scientific applications.	Must sell the MBS approach. (Data availability and security important concerns.)
12. Storage location for source programs	Central and remote station.	Central only.
13. Type of users	Open shop: small, medium and large companies, colleges and universities.	Closed shop: small companies, predominantly.
14. Identity of operators for remote terminals	Open shop: engineers, scientists, secretaries.	Closed shop: accountants, clerks, secretaries.
15. Common programs	Common sub-routines.	Packaged programs modified for each user (30 to 60%); balance hand tailored.

System Characteristics	Remote Scientific Computing	Multiple Business Systems (as of October 1961)
16. Data backup	Local: if lost at central, re-read from local.	Central: if lost, data files must be reconstructed from central backup.
17. Data storage	Limited amount held temporarily in core or disk - memory.	Much data held permanently in disk files.
18. Priority order	1. Debugging runs 2. Production runs	1. Production runs 2. Debugging runs
19. Input - output transmission time	6 minutes/run (input/ output)	3-1/2 - 4 hours/day
20. Use of terminal	Intermittent	3-1/2 - 4 hours/day,average
21. System load	Variable	Less variable than in RSC
22. Distribution of runs (in established system)	Debugging - 33-1/3%* Compilation - 33-1/3%* Production - 33-1/3%* *estimates[36]	Debugging and compilation (during low activity periods) - 10% Production - 90%
23. Program life and modifications	Usually short, with several modifications.	Usually much longer, with monthly modifications.
24. Input - output buffering	Buffering at both central and local.	Central buffering - 90% Local buffering (paper tape) - 10%
25. Processing	Delayed (buffered input/output)	Immediate checking of input Real-time inquiries Delayed (10 minute) processing
26. Programmers	Users (scientists, engineers)	IBM salesmen and systems engineers.
27. Central multiplexing	By complete message or by bit, from a limited number of high-speed lines	By bit from a large number of low speed lines carrying intermittent characters.
28. Security of data	Not typically a problem	A very important problem.
29. System availability	Not critical: a delay of a few hours tolerable.	Critical: should be unavailable no more than 1 to 2 hours/week.

REFERENCES

1. Alt, Franz L., Electronic Digital Computers, Academic Press, Inc., New York and London, 1958.

2. Anthony, Ralston, and Wilf, Herbert S., Mathematical Methods for Digital Computers, John Wiley & Sons, Inc., New York and London, 1960.

3. Polya, G., How to Solve It, Doubleday & Company, Inc., Garden City, New York, 1957.

4. Communications of the ACM, Vol. 4, No. 1, January 1961, cover page.

5. Flores, Dr. Ivan, "An Explanation of ALGOL 60," Datamation, Vol. 6, Nos. 5 & 6, September/October, and November/December, 1960.

6. Woodger, M., "An Introduction to ALGOL 60," The Computer Journal, Vol. 3, No. 2, July 1960.

7. Kelly, John L., et al., "A Block Diagram Compiler," The Bell System Technical Journal, Vol. 40, No. 3, May 1961, pp. 669 - 676.

8. Gordon, Geoffrey, "A General Purpose Simulator," IBM ASDD Commercial Department Report, October 25, 1960.

9. International Conference on Information Processing, Proceedings 1959, UNESCO, Oldenbourg, Butterworths, pp. 122, 126.

10. Hagopian, J. J., "Preliminary Report, Line Speed Buffer," ASDD, San Jose, February 15, 1961.

11. "IBM 7281 I Data Communications Channel," Special Systems Features Bulletin, L 22-6518, October 1960.

12. IBM 7750 Product Description, March 28, 1961.

13. "IBM 7090 DPS, IBM 7281 II Data Communications Channel," Special Systems Features Bulletin, L22-6580, November 1961.

14. Ross, D. T., "Gestalt Programming: A New Concept in Automatic Programming," Proceedings of the 1956 WJCC, AIEE, New York, pp. 5 - 10.

15. SHARE Secretary Distribution No. 84, p. 677.

16. SHARE Distribution No. 964.

17. News and Notices, Communications of the ACM, Vol. 3, No. 4, April 1960, pp. 250 - 252.

18. Bleam, Harvey A., "Some Comments on Remote Scientific Computing," IBM Systems Research Institute, Term Paper No. 2-04.

19. Lovejoy, C. E., Lovejoy's College Guide, Simon and Schuster Inc., New York, 1959, pp. 52 - 53.

20. Fein, Louis "The Role of the University in Computers, Data Processing and Related Fields," Proceedings of the 1959 WJCC, AIEE, New York, pp. 119 - 126.

21. Fein, Louis, "The Computer - Related Sciences (Synnoetics) at a University in the Year 1975," American Scientist, June 1961, pp. 148 - 168.

22. SHARE Secretary Distribution, No. C-2167, No. SSD-84, Kolence to McCabe.

23. Mauchly, John W., "The Lag in Computer Use," Electronics, April 28, 1961, p. 90.

24. "Computer Design in Europe Advances, Use Lags," Chemical and Engineering News, June 5, 1961, pp. 50 - 59.

25. Freed, Roy N., "A Lawyer's Guide Through the Computer Maze," The Practical Lawyer, Vol. 6, No. 7, November 1960, pp. 15 - 45. (Reviewed in Data Processing Digest, February 1961).

26. Haibt, Lois M., "A Program to Draw Multilevel Flow Charts," Proceedings of the 1959 WJCC, AIEE, New York, pp. 131 - 137.

27. Military Applications Research Program: Proposal, IBM Federal Systems Division, September 1960.

28. Codd, E. F., "Multiprogramming Scheduling," PDL, DSD, IBM TR 00.716, April 29, 1960.

29. Codd, E. F., et al., "Multiprogramming STRETCH: Feasibility Considerations," Communications of the ACM, November 1959.

30. "Conference of University Computing Center Directors," Communications of the ACM, Vol. 3, No. 10, October 1960.

31. Reeves, Roy F., "Digital Computers in Universities," Communications of the ACM, Vol. 3, No. 7, p. 407; Vol. 3, No. 8, p. 476; Vol. 3, No. 9, p. 513; and Vol. 3, No. 10, pp. 554 - 545.

32. "G-20 at Carnegie Tech," Computing News, Vol. 9, No. 9, May 1, 1961, pp. 2 - 3.

33. Gotlieb, C. C. et al., "Free Use of the Toronto Computer, and the Remote Programming of It," Computers and Automation, Part 1, Vol. 5, No. 5, May 1956, pp. 20 - 25; Part II, Vol. 5, No. 7, July 1956, pp. 29 - 31.

34. "Transcode Manual," A System of Automatic Programming for FERUT, the Ferranti Mark I Electronic Digital Computer at the University of Toronto, Computation Center, University of Toronto, October 1955.

35. Lasswell, Harold D., "The Social Consequences of Automation," 1958 Proceedings of WJCC, March 1959, pp. 7 - 10.

36. "Progress Report No. 8: Research and Educational Activities in Machine Computation by the Cooperating Colleges of New England," Computation Center, Massachusetts Institute of Technology, Cambridge, Massachusetts, January 1961.