PROCEEDINGS OF THE MEETING

OF THE EASTERN REGION OF THE

1620 USERS GROUP

The Carillon Hotel

Miami, Florida

May 9, 10, 11, 1965

TABLE OF CONTENTS

PAPERS TO BE PRESENTED AT
1620 USERS GROUP MEETING
MIAMI - CARILLON HOTEL
MAY 9, 10, 11, 1965

Session A - 1

Session B - 1

Session C - 1

Page

PROGRAM TEAM SESSIONS

Session A - 2

*Not available at time of publication.

P R O G R A M

1620 USERS GROUP MEETING

CARILLON HOTEL

Miami Beach, Florida
May 9, 10, 11, 1965

SUNDAY - MAY 9

6:00 p.m. - 8:00 p.m.	REGISTRATION	
8:00 p.m. - 10:00 p.m.	SOUND-OFF SESSION	Medallion Room
	Chairman: Dr. James R. Oliver	
11:30 p.m.	SHOW	Cafe Le Can Can

MONDAY - MAY 10

7:30 a.m. - 8:30 a.m.	REGISTRATION	Burgundy Foyer
8:00 a.m. - 9:00 a.m.	NEW USERS BREAKFAST	Sugar 'n Spice Room
9:00 a.m. - 12 Noon	OPENING SESSION	Carillon Room
	Chairman: Dr. Oliver (1) Official opening and welcome by President J. R. Oliver (2) Users Group announcements (3) IBM announcements	
10:00 a.m.	COFFEE BREAK	Upper Lobby
10:30 a.m.	"1130 Computer" D. N. Leeson	Carillon Room
12:00 noon - 1:15 p.m.	LUNCHEON Speaker: Mr. Rudolf Meyer "Education and the 1620"	Burgundy Room
	SESSION A - 1	Carillon Room
1:30 p.m. - 2:00 p.m.	Paper "Unattended Operations of 1620" C. S. Schrodel	
2:00 p.m. - 2:20 p.m.	Paper "Time Sharing on 1620" E. W. Channen (Inter.)	

2:20 p.m. - 2:50 p.m. Paper "The Compleat FORTRAN"
J. A. Haag
(General Interest)

2:50 p.m. - 3:10 p.m. Paper "Undefined Variable Detector for FORTRAN II"
T. J. Scott
(General Interest)

3:10 p.m. - 3:30 p.m. Paper "Music Programs as an Aid to Debugging"
P. J. Jutsum
(General Interest)

SESSION B - 1 — Dominion Room

1:30 p.m. - 2:20 p.m. Paper "Financial Evaluation by Discounted Cash Flow"
Leslie Parker
(Inter.)

2:20 p.m. - 2:50 p.m. Paper "CD4-PDQ (Disk PDQ)"
G. Lilly
(Inter., Systems Programmers)

2:50 p.m. - 3:10 p.m. Paper "1620 Support for a Document Writing System"
C. M. Thatcher
(Inter.)

3:10 p.m. - 3:30 p.m. Paper "UW-M 1620 WISAL" (new language)
C. MacKenna
(Inter.)

SESSION C - 1 — Empire Room

1:30 p.m. - 2:00 p.m. Paper "Use of 1620 Mod I for Simplification of Boolean Algebra Expressions"
C. Plesums

2:00 p.m. - 2:30 p.m. Paper "Plot a (SPS) Block Diagram"
Mrs. S. Young
(General Interest)

2:45 p.m. - 3:15 p.m. Paper "Assembling SPS I Programs with Monitor I"
T. I. Markland
(Inter.)

3:30 p.m. - 4:00 p.m. COFFEE BREAK — Upper Lobby

PROGRAM TEAM MEETINGS:

4:00 p.m. - 5:30 p.m. (1) ELECTRONICS PGM TEAM — Empire Room
A. Spitalny, Chairman

Paper "1620 Back Panel Wiring Program"
Peter Dunham

Paper "1620 Electronic Circuit Analysis Package"
Herbert Wall

Paper "Computer Aided Design of Integrated Circuits"
A. Spitalny

Paper "D. C. Automatic Electric Network Analysis"
H. Pachon
(1/2 hr. Adv.)

4:00 p.m. - 5:30 p.m. (2) STRUCTURAL & CIVIL ENGINEERING PGM TEAM — Dominion Room
T. J. Scott, Chairman

Paper "Gimmik-free Adjustment of Survey Traverse"
R. L. Kenngott
(20-30 min. Inter.)

Paper "3-Dimensional COGO"
J. W. Madden
(45 min.)

Paper "Design of Single, Double, Triple Cell Box Culverts"
Leno Morris
(30 min.)

4:00 p.m. - 5:30 p.m. (3) PROCESS & CHEMICAL ENGINEERING PGM TEAM — Sales Board Room
C. S. Schrodel, Chairman

4:00 p.m. - 5:30 p.m. (4) OPERATIONS RESEARCH PGM TEAM — Carillon Room
J. R. Oliver, Chairman

Paper "CPM & PERT System for Disk and Printer"
A. D. Johnson
(20 min. Inter.)

5:30 p.m. - 6:30 p.m. COCKTAIL HOUR — Sun Terrace

(Compliments of The Carillon Hotel)

6:30 p.m. LUAU — Sun Terrace

9:30 p.m.	EXECUTIVE COUNCIL MEETING	Dr. Oliver's Suite No. 1604
9:30 p.m.	RECENT FILMS of INTEREST to 1620 Users	Empire Room

TUESDAY - MAY 11

SESSION A - 2 — Carillon Room

9:00 a.m. - 9:15 a.m. Paper "Fitting Straight Lines to X-Y Data When Both Variables are Subject to Error"
F. K. Durkam

9:15 a.m. - 9:30 a.m. Paper "Use of Orthogonal Polynomials for Curve Fitting"
S. S. Kuo
(Inter.)

9:30 a.m. - 10:00 a.m. Paper "Curve Fitting by Finite Sums of Exponentials"
R. Meshelof
(Adv.)

SESSION B - 2 — Dominion Room

9:00 a.m. - 9:30 a.m. Paper "TYGO - Load & Go - SPS for Paper Tape"
R. C. Read & P. J. Jutsum

9:30 a.m. - 10:00 a.m. Paper "SPAS - Load & Go SPS for 40-60K Card System"
G. G. Billin
(Adv. - Inter.)

SESSION C - 2 — Empire Room

9:00 a.m. - 10:00 a.m. WORKSHOP:
"Understanding FORTRAN II"
(FORTRAN II sub-programs written in SPS)
L. Hoffman
(Adv.)

10:00 a.m. - 10:30 a.m. COFFEE BREAK — Upper Lobby

SESSION A - 3 — Carillon Room

10:30 a.m. - 11:00 a.m. Paper "Fitting of Non Linear Regression Functions by Least Squares"
Jerry Kemp
(Inter.)

		SESSION B - 3	Dominion Room
10:30 a.m. - 10:50 a.m.		EDUCATION TEAM SESSION	
	Paper	"Planning & Running Short Term Workshop on Comp. Orient. & Prog." Mrs. Joyce Fodor (General Interest)	
10:50 a.m. - 11:20 a.m.	Paper	"A Ten Year Budget Projection for Educational Institutions" J. A. Ferling (Elem.)	
		SESSION C - 3	Empire Room
10:30 a.m. - 12:00 noon		WORKSHOP CONTINUATION L. Hoffman	
		SESSION A - 3	Carillon Room
11:00 a.m. - 11:30 a.m.	Paper	"Autolog Digiklok Mod for 1620 Monitor I System" E. F. Staiano, D. E. Atkins (Inter.)	
11:30 a.m. - 12:00 noon	Paper	"Disk Version of FORTRAN II Incremental Plotter Subroutine" P. Larrea (Plotter users)	
		SESSION B - 3	Dominion Room
11:20 a.m. - 11:50 a.m.	Paper	"Test Scoring System" D. G. Owen	
12:00 noon - 1:15 p.m.		LUNCHEON	Burgundy Room
1:30 p.m. - 3:00 p.m.		"M.P.L." (Multipurpose Programming Language) D. N. Leeson	Carillon Room
3:00 p.m. - 3:30 p.m.		COFFEE	Upper Lobby
3:30 p.m. - 5:00 p.m.		ANSWER TO SOUND OFF	Carillon Room

SUN OIL COMPANY
RESEARCH AND ENGINEERING
ENGINEERING DIVISION

UNATTENDED OPERATION OF THE 1620 COMPUTER

by

C. S. Schrodel

For presentation at the Eastern Region 1620 Users Group Meeting

May 10, 1965

Miami, Florida

UNATTENDED OPERATION OF THE 1620 COMPUTER

SUMMARY

The unattended operation system was designed to allow running of an IBM 1620 for long periods without need of an operator and without the limitation of the card read and punch hoppers. The system can execute and compile Fortran II-Version 2 programs, using the 1311 Disk Drive as the input-output device for data and programs. Separate programs used by the system are:

1. A program to load the data and source decks onto the disk.
2. A program to load the object programs.
3. A program to control compilation.
4. A program to control execution.
5. A program to dump the output from the disk.
6. Modified compiler and subroutines.

Present requirements for use of the system are a 60 K core, a 1311 Disk Drive, source programs in Fortran II, and programs which do not use core locations 59,000 to 59,999.

USE OF THE SYSTEM

The normal use of the system is to load the data and programs onto the disk during the late afternoon, allow unattended operation during the night, and dump the output from the disk the following morning. In order to properly load the disk, it is necessary to know the following about the disk layout:

1. The data and source decks are loaded two sectors per card from the sector specified upward. The sector address is specified at load time and must be greater than 600 and also an even number.
2. The object programs are loaded three cylinders per program from the sector specified upward. The sector address of the first program is specified at load time and must be a multiple of 200.
3. The program output followed by the symbol tables of the compiled programs are written two sectors per record (card) starting at 00000.

4. The compiler (passes I, II and intermediate output) uses sectors 18800 to 19999.

5. Compiled object decks are written on the disk one sector per card starting at 18799 downward.

Sectors should be specified for the input and programs so that the work can be fit on the disk in sectors 0 to 18799, and so that generated output will not overlap data not yet used. Efficient usage is to load first the programs with the highest ratio of input to output. The starting address of the programs should be 18800 minus 600 times the number of programs. Data followed by source decks are loaded below the object programs. Operating in this manner, it is possible to have the output overlap data already used, have the symbol tables overlap the source decks, and finally have the compiled object decks overlap the programs which had been previously run.

Each data set and source program must be preceded by a card with a record mark in column one. If it is desired that the same program be reloaded for the next data set, any digit should be placed in column two of the record mark card. For example, if the same program is to be used for three data sets, then the first two data sets must be preceded with cards having a record mark in column one and a digit in column two. A card with a double record mark is placed after the last source deck as an indication to stop. It should be obvious that data and object programs must be loaded in the same order and that compilations are last. The system cannot be used to compile and execute directly.

"LOAD DATA" PROGRAM

The "Load Data" program is loaded into core, followed by the data and source programs. The ADD tables must be in core. During loading, the program types the message DATA START and reads from the typewriter the sector address at which the data is to begin. This location is recorded on the disk in sector 00598. The data is then read in and written on the disk, two sectors per card, starting at the specified location.

"LOAD PROGRAM" PROGRAM

This program is used to load onto the disk those programs which are to be executed. After loading, the program types the message PROGRAM START and reads from the typewriter the sector address from which the programs are to be stored. This location is recorded in sector 00599, and the program stops. The first program to be executed is then loaded into core, together with its subprograms and subroutines. After loading, a branch to 59000 causes the program to be written onto disk at the proper location. The second program is then loaded into core, and the process repeated.

To prevent the possibility of check stops due to undefined variables, it is convenient to have the core cleared to flagged zeros. However, it is necessary that the "Load Program" routine in the 59,000's not be cleared. The proposed means of doing this is to clear the core to

flagged zeros, restore the add tables, and record this "clear core" in sectors 0 to 00590 before the initial load. This "clear core" is then to be read into core between each loading for a fresh start.

"EXECUTION" PROGRAM

This routine is essentially a monitor and controls the execution of the object programs. The program is loaded into core when the unattended operation is to be initiated and resides in 59000 to 59900. This routine operates in conjunction with a modified subroutine deck to perform the following functions:

1. Load object programs from disk into core.

2. Read through data cards until a record mark card is found, indicating a new data set.

3. Read or write a record on disk whenever a card would have been read or written by an unmodified subroutines set.

4. Terminate execution and go to the next job whenever an error F7 occurs, or 25 errors other than F7 and F8, or a record mark card is read for a data card.

5. Type OLAP and stop if an input record is to be read from a sector lower than the current output sector.

6. Write a record between jobs to control the dumping of the output.

This routine reads a program into core locations 0 to 58999 whenever a branch to 59000 occurs. After loading the program, it reads through the input records until it finds a record mark card. If the record mark card does not contain a digit in column two, it increments the program counters by 600 so that the next program will be read into core on the next branch to 59000. The counters are not incremented when column two contains a digit. The routine then types START and executes a 4900409.

"COMPILE" PROGRAM

This routine works in conjunction with a modified compiler to allow compilation without use of the card reader or punch. Compiler, source statements, intermediate output, symbol table, and object decks are all read from or written on the disk. This program is loaded on the disk by the "Load Programs" routine following the last program to be executed. After termination of the executions, this program is read into core and takes control. The routine requires about 2000 core positions. Functions of this routine are as follows:

1. Read in Pass I or Pass II at the appropriate time.

2. Read or write source statements, intermediate output or object deck using disk instead of cards. Intermediate output starts at sector 19171 for each job.

3. Write symbol tables statements on the disk rather than the typewriter.

4. Type OLAP and stop if source statements are to be read from a higher sector than object deck is to be written.

5. Write a control record for use of "Dump Disk" routine between symbol tables and object decks.

6. Between jobs, read through source statements until a record mark card is found. Stop if it is a double record mark.

"DUMP DISK" PROGRAM

This routine is used to dump the results of the unattended operation from disk onto cards. One alphameric card is punched for each two sectors starting at sector 0 and continuing through the number of sectors written for output and symbol tables. The program then punches out all compiled object decks one card per sector, starting at 18799 and continuing down through the number of records written for object decks. The program stops after punching the output of each job to allow separation of cards. Depressing start causes a continuation to the next job. After punching the last record written, the program types the message END.

MODIFIED COMPILER

Changes were made to both passes of the Fortran II-Version 2 compiler. The changes are as follows:

1. Every card read or written has been converted to a branch into the "Compile" program for the reading or writing of disks.

2. All halts have been removed. Those which had been terminal now generate a branch to the next job.

3. The header card appears only in front of the symbol tables.

4. The use of sense switches is eliminated. Symbol tables and traces are standard. Customary batch compiling is eliminated. Source listings are not available. Subroutines are not punched.

5. PAUSE, STOP, and END statements generate a 275900059899 instruction instead of a 48. This is to prevent undesired halts during unattended operation. If desired, a halt and branch back can be placed at 59000 for attended operation.

MODIFIED SUBROUTINES

A modified subroutine deck is used for unattended operation. The subroutines branch into the "Execution" program for input and output

and the listing of the error message other than F8. Error F8 messages have been modified so that they appear only on the typewriter.

C. S. Schrodel

C. S. Schrodel

APPENDIX: OPERATING TECHNIQUE FOR UNATTENDED OPERATION

Load Data

1. Load add tables if core has been cleared to zero.
2. Load "Load Data" Program.
3. Type sector address at which data is to start; must be greater than 00600 and must be an even address.
4. Stack data followed by source programs, and read into core.
5. Each data set and source program must be preceded by a card with an 028 punch in column one. This card must have a punch in column two if the same program is to be reloaded for the next data set. The last card must have an 028 in both columns one and two.

Load Programs

1. Clear core with a 31$\bar{0}$000300002 command.
2. Load add tables.
3. Store this clear core on the disk with the "Load $\bar{0}$ on Disk" routine.
4. Load "Load Programs" routine.
5. Type sector address at which programs are to begin. Must be a multiple of 200. Normally, 18200-600 times the number of programs to be executed.
6. Load 1st program to be executed and branch to 59000. The Version 2 unattended subroutines must be included.
7. Clear core with the "Read $\bar{0}$ from Disk" routine.
8. Repeat 6 and 7 for each program.
9. Load "Compile" Program.

Execution

1. Load add tables if core has been cleared to zero.
2. Load "Execution" routine.

Dump Disk

1. Load "Dump Disk" routine.
2. Remove cards from punch hopper.
3. Press START.
4. Repeat 2 and 3 until END is typed.

In case processing was interrupted because of error, proceed as follows:

1. Read $\bar{1}$8000 into 36825-36829 and $\bar{0}$0000 into 37657-37661.
2. Execute steps 1-3 above.
3. Branch to 59824 after the last symbol table has been punched to punch object decks.

When dumping the disk using this latter procedure, the operator must use the typewriter sheet to know what output is being punched. One job is punched each time the start button is pressed. Do not count compilations which are Pass I only.

If the contents of 36825-36829 and 37657-37661 are recorded after execution, the normal dump procedure can occur anytime simply by restoring the recorded values.

Set-Up Example

Five programs are to be executed and three programs compiled. The second program is to be run four times.

Program 1 has 1000 cards input and 1500 output,

Program 2 has 50 cards input and 250 output for each data set.

Program 3 has 100 cards input and 500 output.

Program 4 has 10 cards input and 100 output.

Program 5 has 10 cards input and 150 output.

Total output = 3250 cards.

Total input = 1320 cards.

Load programs in order of increasing ratio of output to input (Lowest output/input first). Output of the last can safely overlap input of the next to last (provided last output is greater than next to last input).

Data Start = 2(3250-1310) = 3880

Program Start = 18200-5(600) = 15200

Data should be stacked as follows:

Record mark card followed by Program 1 input.

Record mark card with a digit in column two followed by first data set of Program 2.

Record mark card with a digit in column two followed by second data set of Program 2.

Record mark card with a digit in column two followed by third data set of Program 2.

Record mark card followed by fourth data set of Program 2.

Record mark card followed by Program 3 input.

Record mark card followed by Program 4 input.

Record mark card followed by Program 5 input.

Record mark card followed by first source program..

Record mark card followed by second source program.

Record mark card followed by third source program.

Double record mark card to stop.

Note that compilations are last.

(a) Title and Author: Time Sharing on the 1620

by

Eric W. Channen,
Department of Chemistry,
University of Windsor.

(b) Company and Users Code:

University of Windsor, User No. 7042.

(c) Time Required for Presentation: 20 minutes

(d) Special Equipment Required: Any 1620 using Monitor I

(e) Technical Level of Presentation: Medium

(f) Audience for whom it is intended: Computer Center Staff

(g) ABSTRACT

The Monitor I system has been adapted to permit routine computation to proceed automatically between jobs, thus utilizing idle computer time. The routine or lengthy computation can be interrupted at any time, by the use of a console switch, to allow a short job to be run, and will automatically be resumed when the interrupting job is complete. The effect is accomplished by storing the contents of memory on disc, together with a restart address. The system should be particularly valuable at installations where routine data processing tends to conflict with an open-shop operation.

COMPREHENSIVE FORTRAN PROGRAMMING-A NEW TEXTBOOK

James N. Haag
Purdue University
Indianapolis, Ind.

1620 Users Group Miami Meeting, May 9-11, 1965

I. Introduction
II. Historical Background
III. Computer Technology Curriculum
IV. Compiler Language Programming (FORTRAN)
V. Design Criteria For The Textbook

ABSTRACT

This talk will deal with the arrangement, contents, and underlying philosophy of a new book on computer programming in the FORTRAN language. The pertinent portion of the preface from the book is given below:

"This textbook is intended as an introduction to computer programming for those interested in one of the so-called BEST-SAM areas: Business, Engineering, Science, Technology, Social science, Agriculture, and Mathematics. The mathematical level of this material presupposes only the completion of one year of high school algebra. After mastering this material, a person should be able, independently, to solve many problems by computer methods.

The book is designed to teach the FORTRAN II language. Its objective is to instruct a high school or college student enrolled in a programming course how to program a computer in this language. An adult may utilize this book as a self-teaching device to attain the same objective. No previous experience with a computer is assumed. For the mastery of the material, the availability of a computer is desirable, although not necessary.

The FORTRAN II language given here is that for a 20,000-digit memory IBM 1620 computer with an attached 1311 Disk Drive. This is one of the most widely available computer systems that customarily utilizes the FORTRAN II language. Since there are some two thousand 1620 computers in use, and not all of them have an attached 1311 Disk Drive, this material has been written so that it is applicable also to this situation. Portions of the book which are not applicable to the latter system are denoted with either an asterisk or a statement to this effect. Furthermore, in order to make this book more widely useful, the features of the more comprehensive FORTRAN IV language for the IBM 7090 and 7094 computers are included in an appendix.

The educational philosophy underlying the presentation of the material is to introduce the student to a small portion of FORTRAN II, then have him immediately use this knowledge in writing a complete program before continuing on to another portion of FORTRAN II. A common fault, in this author's opinion, has been avoided by this method.

The entire book may be covered with more intensive work in a two-hour per week one-semester course. A less intensive approach for mastery of the material requires a three- or four-hour per week one-semester course which includes, where feasible, laboratory work on the computer. Many exercises and tested programs are given throughout the book. The following features have been included as an aid to both the student and the instructor:

1) Graded exercises at the end of each chapter.
2) Answers to all even-numbered exercises.
3) Seven complete laboratory exercises with sample computer input and output.
4) 100 review questions and answers.
5) Numerous completely solved FORTRAN II programs.
6) Flowcharting symbols which, unlike older texts, are not inconsistent with the standards defined by the American Standards Association."

I. INTRODUCTION

Textbooks on FORTRAN programming rise and they fall. Their lifetimes vary from months to years. In 1961, McCracken released his book, A Guide to Fortran Programming. In 1963, Organick released his book, A Fortran Primer. In 1965, numerous books on FORTRAN, considerably updated, are being released with deep, emotional, perplexed, fearful expectancies in the hope that another book may join the long-lifetime ranks of these two books. One should note that in history or in economics, a long lifetime may mean 2**4 years, while in our discipline, a long lifetime may mean only 2**2 years.

I wish to tell you of the rise of another textbook on FORTRAN programming as of July 1, 1965. Based on this talk, you should be able to predict when it will fall, and how hard this fall will be. The title of the book is Comprehensive Fortran Programming. Its author is myself, although the contributors consist of several Purdue faculty and numerous men in industry, all of whose names are given in the preface. The publisher, Hayden Book Companies, is optimistic and plans to release the book in two models: an economy paper-bound edition and a luxurious cloth-bound edition. The contents of the book are threefold: FORTRAN with FORMAT for the 1620, FORTRAN II-D for the 1620, and FORTRAN IV for the 7090/7094.

II. HISTORICAL BACKGROUND

In 1962, Purdue University decided to take action based on a Federal report which stated that the then current 1,000,000 jobs related directly to EDP would expand exponentially until 3,000,000 jobs would be available in 1970. By way of background, I might state that most of you, when acting in the category which we call "programmer", are of a special breed called HKE programmers, where HKE is an acronym for "hard knocks of experience". HKE programmers are widely accepted as being quite talented and capable. On the other hand, the system which gives rise to HKE programmers is one of the lowest efficiency systems known to mankind. Years of training, occasionally with random disastrous occurences to man and/or machine, elapse before an HKE programmer arises from the tomb, truly clad in the shining garments of his refined skills. Thank heavens that our medical doctors are not forced, by default, to obtain their training in an analagous manner!

Purdue has already developed, thanks to men such as Sam Conte and Saul Rosen, three degrees in Computer Sciences, a Computer Sciences option for a B.S. in Mathematics, a M.S. in Computer Sciences and a Ph.D. in Computer Sciences. However, graduates of these

programs are destined to be systems analysts, teachers, and researchers, not programmers. Their programming is of a secondary nature in their careers. As a result, Purdue decided in 1962 to immediately set up a 2-year non-college credit curriculum in Computer Technology to efficiently train students in either a commercial or a scientific programming option. Three Model I 1620 computer centers with disks, card input/output, and the full gamut of unit record equipment were installed at Purdue's regional campuses at Indianapolis, Hammond, and Ft. Wayne, Indiana. I at Indianapolis, John Maniotes at Hammond, and later, Maurice Halsey at Ft. Wayne, and about 30 industrial representatives and faculty were given the responsibility of developing the two options' course sequence and content in the 2-year curriculum in Computer Technology.

In the 1962-63 academic year, 100 persons holding down 40-hour-per-week positions started the initial curriculum as part-time students. The classes were all held in the evenings. In the 1963-64 academic year, the revised curriculum was opened to full-time daytime students as well as being continued in the evenings for another 100 part-time students. As of now, over 300 students have started the 2-year program. On the support of all concerned parties, in May, 1965, Purdue converted the entire Computer Technology program into a full college credit curriculum and thus our first graduates this June will receive an Associate Degree in Computer Technology. Also, in May, 1965, Purdue opened up a Junior and a Senior year in Technology to round out the education of those Associate Degree holders who desired to take a B.S. in Technology. Current statistics indicate that of each 100 students who begin this program, about 50 will receive the Associate Degree. It is too soon to say how many of these will go on for the B.S. It is interesting to note that of the entering students in Computer Technology, over 15% already have a B.S. degree.

III. COMPUTER TECHNOLOGY CURRICULUM

The commercial and scientific options of the Associate Degree in Computer Technology each consist of 20 one-semester courses, averaging slightly over 3 credit-hours per course. The objectives of this curriculum and of these courses are listed in Appendix A. The curriculum makes liberal use of the course contents and sequences developed by other universities and the programmer training programs of a number of computer manufacturers and commercial and scientific installations. The course names convey the course contents.

The common technical core of courses for both options is this 7-course sequence of titles: Unit Record Data Processing, Compiler Language Programming (FORTRAN), Machine Language Programming, Assembly Language Programming, Utility Programs, Systems Analysis, and Computer Project. Of the 13 remaining courses, these 5 are common to both options: English Composition, Technical Report Writing, Economics, Algebra, and Statistics. The remaining 8 courses are different for the two options. The commercial option includes two semesters of accounting, computer math, data processing applications, COBOL programming, industrial organization, and two electives. The scientific option includes two semesters of calculus, two semesters of physics, two semesters of numerical analysis, and two electives.

The list of required textbooks and materials for the 13 computer-oriented courses in these two options are listed in Appendix B. This book list changes every four months, that is, three times a year.

The reason for the frequent changes is straightforward: better textbooks are newly published or are brought to our attention. You will recognize among these authors many familiar names: McCracken, Leeson, Germain, and Dodes.

IV. COMPILER LANGUAGE PROGRAMMING (FORTRAN)

Textbook selection is always a difficult procedure. You know from your own experience that no perfect textbook on any subject exists. One must always compromise. Each author, including myself, has either a conscious or an unconscious "Drum to beat", or at least this is what the psychologists claim. Our most difficult textbook selection at Purdue was for the course in FORTRAN, Compiler Language Programming. We have used six different texts and have found them guilty of one or both of these major faults as far as teaching beginning programmers: the "dot-dot-dot" approach and the "fractured flickers" approach. I might add that I do not feel that these two approaches are necessarily faults when instructing intermediate or advanced programmers.

The "dot-dot-dot" approach is one which explains a FORTRAN statement which is directly related to one or more other FORTRAN statements in a context where the related statements are separated by one or more sets of 3 dots. An example of this is the following representation showing the relationship of a DO statement to a CONTINUE statement:

```
      DO 99 J=1,50
        .
        .
        .
   99 CONTINUE
```

Another example of this is the following representation showing the relationship of a FORTRAN IV DIMENSION statement with adjustable dimensions to its subprogram:

```
      SUBROUTINE DOT(...,A,I,J,...)
        .
        .
        .
      DIMENSION ...,A(I,J),...
        .
        .
        .
      DO 22 K=1,I
      DO 22 L=1,J
```

In this latter example, one not only contends with two vertical sets of dots, but with four horizontal sets of dots. Granted, the "dot-dot-dot" approach is excellent for conveying a logical relationship to an intermediate or experienced programmer, but it is ridiculous to enmesh a beginning programmer in a sea of 18 dots as in this latter example. Can you picture an occasional sincere programmer punching this, his first subprogram, into a source deck where the second, third, fourth, sixth, seventh, and eighth cards contain only a period in column 12?

The "fractured flickers" approach is one which explains a FORTRAN statement as an individual entity with little or no regard for the logical relationship of this entity to the rest of the FORTRAN statements in the program. This is by far the more serious error of

omission in textbooks on FORTRAN programming. The more trivial example of this approach is stating that "the PAUSE statement may be placed anywhere in a FORTRAN program except as the last physical statement" and saying no more about the PAUSE statement. Shouldn't one point out that if the PAUSE statement is the next-to-last physical statement in the program, then pushing the START button will not transfer control to any statement in the program?

As far as training beginning programmers, the ultimate example of the "fractured flickers" approach is typified by those textbooks and manufacturers' manuals which present the first complete FORTRAN program, right down to the END statement, on page N, where the text consists of a total of N pages or so. I would hate to train a freshman engineer to design an automobile by spending X hours studying wheels, Y hours studying pistons, and Z hours studying windshield wipers, where the entire car was not mentioned until the last day of the course. If the resulting car didn't just plain collapse at the end of the assembly line, it would probably run sideways with its bumper in the air and the windshield wipers sticking out of the tailpipe.

On the positive side, I will readily admit that there is an important place and a need for clear, precise language specification texts. I might add, and this is a personal opinion, that my contacts with other users' groups have convinced me that IBM does a relatively commendable job in preparing its manuals, both of the reference type and of the programmed instruction type. On the other hand, no doubt many of you know of cases where manufacturer XXX's reference manual stated "so-and-so" and when "so-and-se" was compiled and executed, the result was X unhappy hours of down-time for unclobbering the clobbered configuration.

On the negative side, I will say that very, very, few, in fact, a miniscule percentage of authors have not violated an educational principle which has several thousand years of seniority. It is: "One masters a discipline by learning about a small portion of it and immediately applying this bit of knowledge toward a full solution of a limited problem before learning the next small portion of the discipline." A FORTRAN program is acknowledged by all to be greater than the sum of its component parts, the FORTRAN statements! Consequently, shouldn't a beginning programmer apply each new bit of knowledge to the full solution of a small and complete FORTRAN program before proceeding on the route to the next portion of the language? In physics, and mathematics, and engineering, one always follows this method of fully solving limited problems before proceeding onwards in the text. I think the problem in FORTRAN books, as in all programming books, lies in the fact that our profession is so young and so rapidly changing that no textbook, including my own, fully can capture this time-tested method used in mathematics, engineering, and the physical sciences.

V. DESIGN CRITERIA FOR THE TEXTBOOK

Those of us associated with the Computer Technology program at Purdue University decided that another book on FORTRAN programming should be written for the Compiler Language Programming course. We were not primarily interested in money, fame, or advancement, but just in having a suitable textbook in order to make our job easier. The book, Comprehensive Fortran Programming, was written, and has

been used and rigorously "de-bugged" by several hundred students over the past year and a half. The book was prepared with the point of view that the following occurences should be held to a minimum or entirely eliminated if possible:

1) the "dot-dot-dot" approach;
2) the "fractured flickers" approach;
3) the failure to mention in which statements the rules of FORTRAN commonly change when one writes programs for the computers of other manufacturers or other models of the same manufacturer;
4) the failure to utilize flowcharting symbols which have been defined as standards by the American Standards Association.
5) the failure to include all of the more commonly used input/output devices.
6) the failure to include all of the more commonly used versions of FORTRAN.

In order to avoid the "dot-dot-dot" approach, all of the many FORTRAN programs are given in their entirety as meaningful examples of some calculation often performed on a computer. For example, when the DO statement is discussed, it is shown in many different programs both with and without the CONTINUE statement.

To avoid the "fractured flickers" approach, the first entire, complete, whole FORTRAN program occurs on page 2 of the book. It is:

```
      SUM=2 + 2
      PRINT 1, SUM
    1 FORMAT (F10.0)
      END
```

Note that in this program one has introduced an arithmetic statement, an output statement, a specification statement, and a control statement. Obviously, none of these classes of statements is covered in any great depth in Chapter 1, but immediately the emphasis is on an integrated approach where every FORTRAN statement is treated in the context of an entire program. Note that one may rearrange this program above into 24 possible sequences of the four FORTRAN statements. Thus, Chapter 1 explores which of these 24 sequences are valid and invalid and, inall cases, the reasons for this. The student, for example, can clearly comprehend that this sequence is invalid because the computer"can't output a value for a variable until it has found out what that value is":

```
      PRINT 1, SUM
      SUM= 2 + 2
    1 FORMAT (F10.0)
      END
```

A complete program is given and explained every few pages throughout the book. Furthermore, each chapter is built around a particular program, with the level of complexity gradually increasing until, in Chapter 7, the program consists of over 100 statements, comprising a mainline program and several FUNCTION and SUBROUTINE subprograms. This large number of programs has led to an uncommonly lengthy book, as far as FORTRAN books go, of over 200 pages. In the final chapter, Chapter 8, all of the emphases is on optimizing FORTRAN programs with respect to size, execution time, and generality

As regards the third point above, I might make a brief comment. At my latest count, there were 18 or so different computer manufacturers in the U.S. who routinely could provide a FORTRAN II compiler with one or more of their computer models. Since the FORTRAN II in Comprehensive Fortran Programming is written specifically for the 1620, it was necessary to select a design criterion as to just how to indicate the changes in FORTRAN II as utilized by the 17 other manufacturers and by other IBM computers. Authors such as Organick in the "good old days", back in 1963, could spell out exactly the changes in the language for each of the four or five different manufacturers' computers. Today, unless you wish a book to have, say, 50% of its words enclosed in parentheses or marked by asterisks, this is not readily feasible. Thus, the design criterion was chosen to be this: When a FORTRAN statement's form or usage was different for several computers, this was pointed out and the most common alternative form was usually given. As an example, the book points out that when output statements utilize an E-type specification in 1620 FORTRAN, the value of d in Ew.d must always be less than or equal to w-6, but for the 7090/94 and several other computers, d in Ew.d must always be less than or equal to w-7.

With respect to the fourth point, certainly my comments can shed no light. Many of you have privately and publicly expressed yourself strongly in favor of standards in our rainbow-splattered profession. At one time, if RCA did their flowcharts this way, then GE felt compelled to do their flowcharts that way and so on. As you know, the ASA has adopted and publicized a limited number of flowcharting symbols as standards. The more enlightened manufacturers, including fortunately our own, have included the ASA symbols along with their own more specific symbols in their latest plastic templates. Certain manufacturers and authors are still going their own way, using symbols from the Middle Ages (applied, not to history, but to our profession, this means the 1950's and early 1960's). Comprehensive Fortran Programming uses only the 10 program flowcharting symbols given by the latest IBM template and none of these symbols are currently inconsistent with the ASA standards.

In order to meet the fifth criterion above, the book includes, with explanations and examples, the FORTRAN input/output statements corresponding to these devices: console typewriter, paper tape reader, paper tape punch, card read punch, magnetic disk drive, magnetic drum drive, and magnetic tape drive, and on- and off-line printers. The sixth and final point is quite important in my opinion. Although the three more common levels of FORTRAN are included in the book, that is, FORTRAN with FORMAT, FORTRAN II-D, and FORTRAN IV, it was felt necessary to choose one of these levels for the mainstream of the text. The FORTRAN II-D level was chosen for these reasons:

1) More compilers are available for this level than for the other two levels.
2) This level is intermediate in complexity between the other two levels.
3) This level includes almost all of the FORTRAN language.
4) The 1620 with 1311 is a widely used configuration.

Once this mainstream level was chosen, we needed a design criterion as regards the method to be used to indicate the FORTRAN with FORMAT and FORTRAN IV changes to the language. One of the overall aims in writing this book was to make the material flow smoothly with

no footnotes and very few logical breaks. So it was decided to use asterisks to denote which of the 49 FORTRAN II statements did not apply to FORTRAN with FORMAT and also add a sentence in parentheses when they did apply but with changes. These changes were indicated, for example, by saying that only 4 general forms of subscripts are permitted in FORTRAN with FORMAT instead of 7 general forms as in FORTRAN II and FORTRAN IV. It was decided that the FORTRAN IV language insertions would damage the smooth flow of the text. As a result, Appendix A is a 20-or so-page presentation of FORTRAN IV which is based in its exposition on the 200-plus pages of FORTRAN II material. This method permits one to learn the FORTRAN IV languages in just a few hours after the completion of the body of the book. Recall that one only need learn about the DATA and logical IF statements, the Type statements, logical, double-precision, and complex constants, additional specifications, labeled common, adjustable dimensions, and a few more generalized FORTRAN II statements. The FORTRAN IV which is included is that for the IBM 7090/94. It was included not only to widen the market for the book to SHARE users, but to facilitate the changeover of a number of 1620 users from the 1620 to either the 1130 or the System 360 since their FORTRAN compilers are similare to the 7090/94 FORTRAN IV compilers in certain respects, although not in all respects.

In conclusion, I might remark that it is certainly easier to make a switch to the New Programming Language by way of FORTRAN IV than directly from FORTRAN II. Thus, this Appendix A might help some programmers to bridge the gaps between FORTRAN II and the eventual NEW Programming Language, which keeps changing its acronym from NPL to MPL to MPPL to PL/I and who knows what else. I heard a completely unconfirmed rumor that two more acronyms, or more correctly, names, are circulating in private conversations: "WATSON" and "9-EDGE".

* * * * *

SUN OIL COMPANY
RESEARCH AND ENGINEERING
ENGINEERING DIVISION

UNDEFINED VARIABLE DETECTOR

by

Thomas J. Scott

For presentation at the Spring, 1965 1620 Users Group Meeting

May 9 - 11

Miami, Florida

UNDEFINED VARIABLE DETECTOR

INTRODUCTION

The Undefined Variable Detector program (UVD) is written in S.P.S.-II to be an aid in debugging other programs that are written in Fortran II-Version II. Basically, the "UVD" checks to see that the arithmetic subroutines and the output portion of the I/O subroutines of the Fortran II-Version II package do not get an opportunity to manipulate improperly defined fields. A total of nineteen of these subroutines are altered by the "UVD", in order that this field definition check may be made in every case where control is passed to a subroutine that might, in normal operation, encounter an undefined variable. A list of these altered subroutines may be found on page 2.

The "UVD" is loaded begind the Fortran II-Version II program at object time, and in no way affects the results normally obtained from the program.

GENERAL DISCUSSION

Undefined variables do not always indicate their presence in the same manner. The most obvious means of detection is the sudden and shocking appearance of the check-stop light on the console. The displaying of the contents of IR-2 will indicate the return address from the subroutine package, but this does not always define the portion of the program in which the undefined variable was encountered. Depending upon the length of the field transferred, the program may have been terminated abruptly, or it may have solved several problems before feeling the effect of the undefined variable. The latter case may create a tedious searching problem for the programmer.

The most potentially dangerous situation, however, is the one in which the undefined variable remains undetected.

The "UVD" will identify an undefined variable the instant it is encountered, and will type a message indicating its symbol table location. the programmer may then define the variable via the typewriter, and return control to the "UVD."

CORE REQUIREMENTS

The program requires 1200 core positions. It utilizes the indirect addressing feature. It may be re-assembled at any location in core simply by changing the "define origin" card at the very beginning of the source deck.

TIMING

Sample runs have indicated an increase of approximately 50% in run time. This is based on a Mod-1, without the floating point hardware feature.

SUBROUTINES ALTERED BY THE UVD

Subroutine Abbreviation	Core Location
TOFAC	01314
FLAD	02466
FXAD	01506
FLSUB	03038
FXSUB	01538
FLMUL	03082
FXMUL	01632
FLDVD	03254
FXDVD	01676
RFLSUB	02372
RFXSUB	01562
FLRDVD	03470
FXRDVD	01762
RSNFL	02372
RSNFX	02372
FLEXP	04586
FXEXP	03996
I/O	05938
FIXI	03574

Title: Music Programs as an Aid to Debugging

Authors: Ronald C. Read and Peter J. Jutsum

Direct enquiries to: Ronald C. Read
Computing Centre
Mathematics Department
University of the West Indies
Mona, Kingston 7, Jamaica.

The problem of debugging a new program can at times be very tedious and frustrating, and any aid to the operator to enable him to follow what is going on in the computer when he is trying to find a fault in a program must be of some value. The construction of the 1620 enables the state of certain of the registers to be examined with the aid of the indicator lights, but these are of very little value except in the static case with the machine halted. This halt can be made manually, but frequently in the course of debugging it is a check stop, and either way, the addresses of the halt does not always lead very rapidly to the cause of the fault. It is the purpose of this paper to suggest that the radiations from the core storage of the computer, which can easily be made audible, can be a valuable tool to aid the programmer in the debugging stage.

A small radio set is all that is needed in the way of extra equipment, and if this is placed on top of the machine immediately above the program switches, and tuned away from a broadcast transmission but as close to the clock frequency as possible, this is sufficient to hear all that is needed. (The clock frequency of the Model i is 1 Mc/s). The clock pulse is modulated by the arithmetical and other operations and this, when demodulated by the radio receiver, gives a noise in the audible range.

The sounds picked up by this method vary considerably according to the nature of the operation that is going on at the time, and it requires a certain amount of practice to recognise the sounds and interpret them in a useful way. To gain a little experience, it is quite a good idea to "listen" to some programs with which one is familiar.

However some of the noises are very characteristic, particularly fairly tight loops, and if these occur in the program, either intentionally or because of some fault of programming, they are very easily recognised. The dismal sound of clearing memory is one of those that cannot be missed!

Consider as an example a table look-up section of a program that is fairly common, such as:-

```
        TFM   *+18,START
LOOK    C     , SYMBOL          960
        BE    OUT               160
        AM    LOOK+6,10,10      560
        CM    LOOK+6,END        560
        BNE   LOOK              200
                                    et cetera.
```

This has an execution time for the cycle, assuming that the entries in the table are all of ten digits and positive, of 2,440 microseconds, and this means that a note of frequency 1,000,000/2,440 c.p.s. will be generated. (The timings are for the Model 1) i.e. about 410 c.p.s. or about G above middle C.

Less pure notes, or perhaps they should be called noises, come from loops whose execution times are not constant. For example, in the loop above, if the stored entries in the table are not all of the same length, and not all positive, the instruction LOOK will not have a constant execution time, but it will vary between 960 and 280 microseconds. The noise of this loop will have fundamental frequency components between 410 and 570 c.p.s., but also many harmonics, and the result is not at all musical.

(Demonstration on tape recorder of these two sounds)

Parts of a program that are repetitive in this sort of way can usually be fairly easily recognised, and even if no more is done than to listen to the new program on first running it, this can save a considerable amount of searching if an exasperating check stop is encountered. At the very least it can save the time involved in running a trace routine from the very beginning of the program when in fact the fault does not occur until a good deal later.

The next demonstration is of a program that has several different table searching routines in it. It is in fact an S.P.S. processor that is the subject of a later paper. The loops that can be heard are, first the clearing of the input area, secondly the looking up of the OP code in a table, (this is sometimes very brief, and is apparently missing) and then the looking up of the symbol or symbols that may be in the operands or in the label. This last is an irregular lookup routine, in that the stored symbols are of varying lengths.

(Demonstration of the sound of TYGO allowing it to run on to the symbol table revision)

The final sounds which are not typical of the rest are those of a routine which eliminates the no longer needed parts of the symbol table and then repacks it by a digit by digit transmission. This happens only rarely in the program, but points to another valuable use of the listening technique. The repacking process just heard takes a considerable time, and in an early version of the program that worked without errors, it was being operated too often. On the evidence that came from listening to the program, it was rewritten in this part, and

the result was a considerable saving of time. I do not contend that the wasting of time would not have been discovered eventually anyway, but the impact of hearing the wasted seconds squeaking away made the decision to go through the agony of rewriting easier.

Another program in which the listening technique has been useful is a language translation program. This has a very characteristic sound, of a more than usually musical nature in parts.

(Demonstration of the sound of the German translation program when working correctly)

The initial noise is the looking up of the words of the input sentence in a dictionary whose entries are of variable length. The more musical part is the sound of looking up in lists of two, three and four word groups. These lists are each of a constant length of entry, but they differ between lists. (I must emphasise that the hunting horn sound was not intentional!!). This process enables matters of syntax and word order to be sorted out.

During the debugging stage in this program, a check stop occurred with some but not all input sentences. Listening to the sound in this case leads very quickly to the fault.

(Demonstration of the German translation program with a record mark missing)

The word lookup is obviously functioning properly, and the looking up of the word groups starts well, but degenerates into the noise associated with looking up in a list of irregular length entries. The cause of the fault was found to be that the terminal record mark that ends the search in one of the syntax lists if an entry is not found, was missing.

All the examples that I have given so far have been of programs that have fairly tight loops as a proper part of themselves, and this is by no means always the case. It is certainly true that not much can be gleaned from the sound of other parts of the program since the capacity of the ear to sort out very irregular sounds is very limited, but the listening technique is not defeated in these cases. It is recommended practice to introduce a number of unnecessary halts into a program, subsequently to be replaced by NOP instructions or to be left out altogether on recompiling, and these serve to enable the operator to isolate the section in which a fault occurs. It is suggested that instead of the halt, the inserted instructions are BTMs to a subroutine that generates a characteristic note, and that these are later replaced by NOPs. While the BTM instructions remain however, the fact that the program has got to certain points without mishap can be clearly recognised by listening to it. The duration of the inserted notes can be varied, and they serve a triple purpose. They furnish audible progress reports, they slow down the program to a speed that enables the operator to keep track and they can be made of sufficient duration in crucial cases to enable the program to be halted at a desired point. A basic subroutine (using indirect addressing) of only 43 digits follows. If indirect addressing is not available a slightly .....

longer subroutine is needed

```
SING  TR   SING-1,SING-1,611
      SM   *+9,4,710
      BNE  SING
      BB
```

and the linkage to be introduced into the program in place of the traditional halt

```
      BTM  SING,xxxxx
```

where xxxxx is an address somewhere in the arithmetical tables. If the Q operand is 400, this will produce the highest possible note of a fairly short duration. Lower addresses will produce progressively lower notes of progressively longer duration. If an operator wishing to try this would be offended by notes that are not in harmonic relation to one another, I will gladly supply a list of suitable Q operands together with their musical equivalents.

The next recording is of a program that is searching for closed circuits in an arbitrary network. In the first part of the recording there is no modification made, and you can hear that it would be very difficult to sort out the various operations that are going on. In the later part of the recording, we have the same program, but it has had some musical "telltales" introduced into it, and knowing the points in the program where these have been introduced, and the pitch that corresponds with each mark, one can follow the progress of the program. When that recording comes to an end, it is because of a check stop (deliberately fixed), and it is possible to tell if we listen attentively to the sounds, that it occurs after the first incidence of Low D. This enables us to isolate the fault immediately.

(Demonstration as described above)

One can go further in this, and by writing a slightly longer subroutine in place of SING, it would be easy to arrange that the note played at a point in the program depended on the state of a switch which is being set. For instance, it sometimes occurs that a program fault is caused by the removal or setting of a flag or record mark which is not intended by the programmer, and on those occasions where this doesn't happen until the program has run for some time, chasing the bug can be very tedious. It would be an idea to put in an audible warning of this, long enough to wake up the operator.

There is one rather risky aspect of this technique. In time, the fascination of the noises made by a program becomes too great, and the intervening calculations wane in their importance. The result could well be

(Demonstration of a short excerpt from "The Flight of the Bumble Bee" by N.Rimsky-Korsakov.)

FINANCIAL EVALUATION PROGRAM

Leslie Parker
Manufacturing Engineering
Poughkeepsie, New York

Management decision making should be an enlightened process. Lack of time and useable information, however, often conspire against it. The decision maker may be forced to rely more heavily on his own judgment than he would like.

One factor which should enter into every business decision is the financial justification for the change. The Financial Evaluation Program can provide this information. Too often the process of financial evaluation involves:

- Gathering cost data by someone technically aware of the situation.
- Explanation of this data to someone with a financial background (with a resultant loss due to differences in semantics, purposes, and backgrounds).
- Lengthy financial analysis, often by non-standard, manual methods, and with a poor to cursory understanding of the problem.
- Explanation of the results of the evaluation to the decision maker, through the eyes of the person doing the calculations (again with losses due to differences in semantics, purposes, and backgrounds).
- Decision made on lost premises, and an incomplete understanding of the meaning and methods of the financial manipulations.

The shear length of time required for the analysis often prohibits its use, and in those instances where the time is available, the evaluation suffers in the translation from analyst to evaluator to decision maker.

This program by comparison is a standard tool which the analyst uses to abstract and compress financial data. The evaluation is swiftly performed, and placed in a form familiar and understandable to the decision maker.

Following are some of the reasons for having economic evaluation:

1) SUSTAIN AND INCREASE PROFITS OF FIRM
2) PERMIT UNIFORM JUSTIFICATION OF ALL PROJECTS
3) ENCOURAGE CONSIDERATION OF ALTERNATIVES
4) REDUCE BIAS OR IRRATIONAL APPROACHES
5) CURTAIL FALSE STARTS
6) PERMIT PERIODIC FINANCIAL AUDITS
7) ENABLE MANAGEMENT TO MORE EQUITABLY EVALUATE ALTERNATIVES
8) PROVIDE MEASUREMENT OF UNCERTAINTY

The following list details some of the specific attributes of the Financial Evaluation Program.

1) FLEXIBLE
2) PRACTICAL
3) CONSISTENT
4) COMPLETE TREATMENT OF COSTS
5) COMPUTERIZED
6) DISCOUNTS MONEY
7) RECYCLES DEPRECIATION
8) RECOGNIZES CASH VS. ACCOUNTING COST

The Discounted Cash Flow technique is illustrated in Figure #1. First a Net Cash Outlay is calculated for each alternative method of implementing the project, for each year of the financial life (NCO-A and NCO-B). From these values for each year, a net cash difference is calculated (CASH FLOW). OL stands for outlay, analogous to Investment (INV) and FB stands for flowback, analogous to savings (SAV). The INV or SAV column is an adjusted Cash Flow, in which the deferred investment (OL which chronologically follows FB) have been removed through a Cost of Capital adjustment. From the INV or SAV column, the compound interest rate is calculated which would have to be acquired to realize equivalent return from a bank, by depositing the investments. This interest rate is the Rate of Savings for the evaluation. Finally each entry in the INV or SAV column is discounted to the present date using Rate of Savings as the compound interest rate.

DISCOUNTED CASH FLOW TECHNIQUE

Year	NCO-A	NCO-B	CASH FLOW	INV OR SAV	DISCOUNTED VALUE
1964	5,000	10,000	OL 5,000	INV 5,000	5,000
1965	10,000	5,000	FB 5,000	0	0
1966	5,000	12,000	OL 7,000	INV 1,650	1,355
1967	10,000	5,000	FB 5,000	SAV 327	243
1968	5,000	10,000	OL 5,000	0	0
1969	20,000	10,000	FB 10,000	SAV 10,000	6,111

Cost of Capital is 7.0 %

Rate of Savings is 10.4 %

The following outline describes the Program Method, in a broad form.

COST COMPARISON

To compare two methods of accomplishing the same project, assume both methods will derive the same income but incur different costs; therefore, compare the costs involved.

INPUT PREPARATION

Analyst lists projected costs for each method in each year of project life.

INPUT CATEGORIES

Complete check list of expense categories is provided.

NET CASH OUTLAY

Program calculates Net Cash Outlay for each method, in each year of Financial Impact Life by calculating costs depreciation, and tax credits. (See EXPENSE TREATMENT.)

DIFFERENTIAL CASH FLOW

The difference between the two Net Cash Outlays is found for each year of the Financial Impact Life and printed in the Cash Flow column of the output.

DEFERRED INVESTMENT

All deferred investments are reconciled and a new Cash Flow is generated (INV or SAV column).

RATE OF SAVINGS

Compound interest formula is evaluated to find RATE OF SAVINGS interest rate which would have to be applied INV in order to realize equivalent savings.

PRESENT VALUE

The present value of each entry in the INV or SAV column is calculated using the RATE OF SAVINGS.

EXPENSE TREATMENT

CURRENT INVESTMENT (CI) (Year End Value of Asset)

- Increases in (CI) are added to Net Cash Outlay (NCO) in that year.
- Decrease in (CI) yields an Income Tax credit, subtracted from NCO in that year.
- Carrying cost calculated on Inventory and added to NCO.

FIXED INVESTMENT (FI)

- Each (FI) is added to NCO in year it occurs.
- Each (FI) except land, is depreciated over appropriate tax life, and recycled, if necessary. Income Tax credits are calculated on this depreciation for each year; these are subtracted from NCO.
- Incentive tax credits are calculated for each (FI) except land, and subtracted from NCO in the year of purchase.

INVESTMENT EXPENSE (IE)

- (IE) in manyears are converted to dollars and benefit rate applied.
- (IE) added to NCO.
- In non-recycle problem, Income Tax credits are calculated and subtracted from NCO.
- In recycle problem, expenses are depreciated over tax life of product, Income Tax credits calculated and subtracted from NCO in appropriate years.

ANNUAL CASH OPERATING EXPENSE (ACOE)

- Convert all manyears to dollars plus benefit rate.
- Add all ACOE to NCO.
- (Material and Operating Supplies/Turnover Rate) times carrying cost, added to NCO.
- In non-recycle problem, Income Tax credits are calculated and subtracted from NCO.
- In recycle problem, expenses are depreciated over tax life of product, Income Tax credits calculated and subtracted from NCO in appropriate years.

ANNUAL EXPENSES OR GAIN SUBJECT TO TAX

- DORA - Calculate Income Tax credit and subtract from NCO.
- SAS - Calculate Income Tax credit and subtract from NCO.
- Compare SOFA to WOFA:

  If SOFA is greater than WOFA, calculate capital gain tax on difference and add to NCO.

  If WOFA is greater than SOFA, calculate Income Tax credit on difference and subtract from NCO.

- Subtract SOFA from NCO.
- Subtract INCOME from NCO.
- Income Tax is calculated on INCOME and added to NCO.

The figure on the following page shows an input sheet. One of these is completed for each year of the project life. The input form comprises a list of cost categories, bringing to the attention of the analyst the information pertinent to the evaluation. A carbon copy of this sheet is used directly by keypunching for preparing the card input to the program.

A facsimile listing of the information carried on control cards is shown below. This includes the standard manpower, tax and turnover rates.

COMPANY CONFIDENTIAL

FIXED INVESTMENT CONSTANTS BY ASSET TYPE

	LI	BLD	TE	TL	PE	PEA	NPE	SE	GI	OTH
TAX LIFE	25.00	35.00	7.00	20.00	20.00	6.50	9.25	13.00	18.00	2.25
TAX CREDIT	7.00	7.00	4.67	7.00	7.00	4.67	7.00	7.00	7.00	0.00

DOLLAR RATES PER MAN YEAR

DESIGN	DE-BUG	DIRECT	INDIRECT	IND-TECH	MAINT
2665	1757	3150	2746	1575	3780

MISCELLANEOUS CONSTANTS

INVENTORY TURNOVER RATE A 2.00 B 1.00

INVENTORY CARRYING COST RATE 1.7 PERCENT

EMPLOYEE BENEFITS RATE 2 PERCENT

PRODUCT LIFE FOR RECYCLING 2.00 YEARS

FEDERAL TAX RATE 50.0 PERCENT

STATE TAX RATE 5.0 PERCENT

CAPITAL GAIN TAX RATE 25.0 PERCENT

COST OF CAPITAL 1.5 PERCENT

IBM

ECONOMIC EVALUATION

(ORIGINAL)

PROJECT NO. | YEAR | DESCRIPTION | ECONOMIC LIFE | DATE

COST COMPARISON ☐ OR COST REDUCTION ☐ RECYCLE NO ☐ YES ☐

A OR PRESENT	CURRENT INVESTMENT	INVENTORIES	PREPAID INSURANCE	PREPAID RENTAL	OTHER		
	FIXED INVESTMENT	LAND	LAND IMPROVEMENTS	BUILDINGS	TEST EQUIPMENT	TOOLING	PRODUCTION EQUIPMENT
		PRODUCTION EQUIPMENT ACCESSORIES	NON PRODUCTION EQUIPMENT	SPECIAL EQUIPMENT	GENERAL INSTALLATIONS	OTHER	
	INVESTMENT EXPENSE	DESIGN (MAN YRS)	PROTOTYPE	DEBUGGING (MAN YRS)	TRAINING	REMOVAL REARRANGEMENT & INSTALLATION	OTHER
	ANNUAL CASH OPERATING EXPENSE	DIRECT LABOR (MAN YRS)	INDIRECT LABOR (MAN YRS)	INDIRECT TECH. NICAL SUPPORT (MAN YRS)	MAINTENANCE (MAN YRS)	MATERIALS	OPERATING SUPPLIES
		RENTAL EQUIPMENT	FLOOR SPACE	REWORK	REBUILD/OR OVERHAUL	VENDOR SUBCONTRACT	EXPENSE TOOLS
		LEASES	TAXES	UTILITIES	OTHER		
	ANNUAL EXPENSE OR GAIN SUBJECT TO TAX	WRITE OFF OF FIXED ASSETS	DEPRECIATION ON RETAINED ASSETS	SPOILAGE AND SCRAP	SALE OF FIXED ASSETS	INCOME	
B OR PROPOSED	CURRENT INVESTMENT	INVENTORIES	PREPAID INSURANCE	PREPAID RENTAL	OTHER		
	FIXED INVESTMENT	LAND	LAND IMPROVEMENTS	BUILDINGS	TEST EQUIPMENT	TOOLING	PRODUCTION EQUIPMENT
		PRODUCTION EQUIPMENT ACCESSORIES	NON PRODUCTION EQUIPMENT	SPECIAL EQUIPMENT	GENERAL INSTALLATIONS	OTHER	
	INVESTMENT EXPENSE	DESIGN (MAN YRS)	PROTOTYPE	DEBUGGING (MAN YRS)	TRAINING	REMOVAL REARRANGEMENT & INSTALLATION	OTHER
	ANNUAL CASH OPERATING EXPENSE	DIRECT LABOR (MAN YRS)	INDIRECT LABOR (MAN YRS)	INDIRECT TECH. NICAL SUPPORT (MAN YRS)	MAINTENANCE (MAN YRS)	MATERIALS	OPERATING SUPPLIES
		RENTAL EQUIPMENT	FLOOR SPACE	REWORK	REBUILD/OR OVERHAUL	VENDOR SUBCONTRACT	EXPENSE TOOLS
		LEASES	TAXES	UTILITIES	OTHER		
	ANNUAL EXPENSE OR GAIN SUBJECT TO TAX	WRITE OFF OF FIXED ASSETS	DEPRECIATION ON RETAINED ASSETS	SPOILAGE AND SCRAP	SALE OF FIXED ASSETS	INCOME	

The following facsimile output contains:

1) Project designation
2) Cost summations for each alternative
3) Differential Cash Flow; INV or SAV, and DISCOUNTED VALUE
4) Calculated RATE OF SAVINGS, and
5) PAYBACK period

SAMPLE PROBLEM

PROJECT NO 0000373-03 FINANCIAL IMPACT LIFE 8 YEARS DATE 112763

RECYCLED COST REDUCTION

	ALTERNATE A	ALTERNATE B
CURRENT INVESTMENT	0	0
FIXED INVESTMENT	18000	36675
INVESTMENT EXPENSE	1114	2880
OPERATING EXPENSE	99739	40412
TOTAL PROJECT COST	118853	79967

YEAR		CASH FLOW		INV OR SAV	DISCOUNTED VALUE
1964	OL	11092	INV	11092	8169
1965	FB	5680	SAV	5680	3081
1966	FB	5333	SAV	4400	1758
1967	OL	947		0	0
1968	FB	7142	SAV	7142	1548
1969	FB	5281	SAV	5281	843
1970	FB	9771	SAV	7975	937
1971	OL	1823		0	0
TOTAL	FB	19345	SAV	19386	

RATE OF SAVING IS 35.8 PERCENT AT A COST OF CAPITAL OF 1.5 PERCENT

PAYBACK IS 4.1 YEARS

Many advantages accrue from the use of the program; some are listed below:

EXPEDITES APPROPRIATION REQUESTS

PROVIDES RAPID RESULTS

COMPARES VARYING INPUT VALUES (Variable Planning)

COMPARES VARIOUS ECONOMIC LIVES

COMPARES VARIOUS CONTROL DATA RATES

INPUT FORM PROVIDES COMPREHENSIVE CHECK LIST

BROAD APPLICATION

REDUCES USER TRAINING

ACCURATE COMPUTATIONS

CONSISTENT TREATMENT

ANNUAL IDENTIFICATION OF COSTS

HELPS AVOID SURPRISES

INCREASES ECONOMIC AWARENESS IN USER

STRENGTHENS USER'S FINANCIAL ARM

SEPARATES COSTS (Relevant - Irrelevant)

SPOTLIGHTS IMPENDING FINANCIAL TROUBLE

As with every technique there are some cautions which must be observed, in order to obtain valuable output. These are listed below:

APPLICATION

EMPHASIS

ILLUSORY SECURITY

INTERPRETATION OF RESULTS

NECESSITATES GOOD JUDGMENT

The program requires the following computer configuration:

1620 CPU/20 K MEMORY

1622 CARD READ PUNCH

1620 FEATURES

- AUTO DIVIDE
- ADDITIONAL INSTRUCTIONS
- READ PUNCH ADAPTER
- INDIRECT ADDRESSING

The prime financial objective of business is to increase wealth. This requires effort in two directions; first, the minimization of cost; and secondly, the maximization of profit. This program is primarily a cost minimization model, using a detailed listing of the project associated costs, however, it may also be used to evaluate profits by entering project incomes as lumped yearly sums.

Using the newest financial evaluation techniques, and data processing capabilities, this program gives better answers to old questions. It is not meant to make management decisions, which must include many factors not considered by the program, i.e. technical feasibility, certainty of estimated input data, concurrent project experience and displacement. It does provide the engineer or analyst with a strong tool for evaluating the economics of projects under consideration, and thus proves a valuable adjunct to the studies of technical feasibility.

APPENDIX I

The following figure illustrates recycle depreciation, which the program can use. This method of depreciation is used by some firms in the rental business. It serves to defer tax credits into the future so they will parallel the income from the product.

YEAR	1	2	3	4	5	6	7	8	9	
Investment Tax Life Depreciation for $100 Investment	40.00	30.00	20.00	10.00						Line 1
Recycle Depreciation for $40 over 6 Years	11.43	9.52	7.62	5.72	3.81	1.90				Line 2
Recycle Depreciation for $30 over 6 Years		8.57	7.14	5.71	4.29	2.86	1.43			Line 3
Recycle Depreciation for $20 over 6 Years			5.72	4.76	3.81	2.86	1.90	0.95		Line 4
Recycle Depreciation for $10 over 6 Years				2.86	2.38	1.91	1.43	0.95	0.47	Line 5
Total Recycled Depreciation Values	11.43	18.09	20.48	19.05	14.29	9.53	4.76	1.90	0.47	Line 6

NOTE: The $100 investment was first depreciated over a four-year tax life of the investment (Line 1); then each yearly amount from Line 1 was depreciated over a six-year tax life of the product (Lines 2 through 5). These values were accumulated in Line 6, which represents the total yearly recycle depreciation values.

Recycle Depreciation (Dollars)

APPENDIX II

GLOSSARY

CASH FLOW
A chronology of the yearly total Outlays and Flowbacks or Investments and Savings of cash.

CONVENTIONAL CASH FLOW
Cash Flow with Deferred Investments reconciled.

COST COMPARISON
Evaluation method in which both projects are under tentative consideration.

COST OF CAPITAL
The interest rate at which the company can obtain the use of capital.

COST REDUCTION
Evaluation method in which the base project (A) is presently in use.

DEFERRED INVESTMENT
Outlays which chronologically follow Flowbacks in the Cash Flow.

FLOWBACKS (FB)
Differential receipt of cash because of implementing project (B) instead of project (A).

FULL DISCOUNTED CASH FLOW
Present values of all Cash Flow Investments and Savings (Cash Flow discounted to present date at Rate of Savings).

INVESTMENT (INV)
Cash Flow Outlays with Deferred Investments reconciled.

NON-RECYCLE DEPRECIATION
Conventional, single depreciation, i.e., sum of the year's digits, straight line, or double straight line declining balance.

OUTLAY (OL)
Differential payment of cash because of implementing project (B) instead of project (A).

RECYCLE DEPRECIATION
A double depreciation method used by some firms in the rental business.

SAVINGS (SAV)
Cash Flow Flowbacks with Deferred investments reconciled.

C4D

An Operating System Built Around PDQ

by

J. Grant*, G. F. Lilly**, F. H. Maskiell***, M. L. McAteer****

ABSTRACT

C4D is a PDQ operating system for the IBM 1620. It provides for batch processing a mixed group of compilation and execution runs. It permits storage of programs and data on one or more 1311 disk drives. Segmentation of programs is possible under C4D. A superior set of diagnostics has been included as part of the system.

---

* Junior Research Mathematician, Jones & Laughlin Steel Corporation
** Research Supervisor, Jones & Laughlin Steel Corporation
*** Computer Supervisor, Pennsylvania Transformer Division of the McGraw Edison Company
**** Research Physicist, Jones & Laughlin Steel Corporation

THE C4D OPERATING SYSTEM

C4D is an operating system built around an extended version of PDQ FORTRAN.[1] It is written for the IBM 1620 with a 1311 disk file, indirect addressing, and the special instructions TNS, TNF, and MF. C4D will also utilize a second disk drive and the 1443 printer if these are available. It is self adapting for any core size.

C4D consists of the following programs:

1. The statement scan routine
2. The diagnostic routine
3. The compiler routine
4. The class A subroutines
5. The relocatable subroutines
6. The executive routine.

The statement scan routine reads each source program statement, removes the blanks, determines the type of statement and stores this information on the disk for later use by the diagnostic and compiler routines. After it has read the entire source program it calls the diagnostic routine from the disk.

The diagnostic routine is a modified version of the 40K FORTRAN II Diagnostician.[2] The modifications accommodate the differences in language between FORTRAN II and PDQ FORTRAN, permit the use of the 40K Diagnostician on a 20K machine, and incorporate the Diagnostician as a routine within C4D. If the diagnostic routine finds errors it punches or prints error messages, lists the entire source program on the 1443 if available and calls the executive routine from disk. If no errors are found in the program the diagnostic routine calls the compiler.

The compiler routine is a modified version of the original C2 processor for PDQ.[2] Error messages have been eliminated as these are now handled by the diagnostician. ACCEPT and IF(SENSE SWITCH) statements have been eliminated as they have no place in an operating system designed for open shop programming and closed shop operation. The ability to segment programs has been added to avoid core size limitation on program length. More symbols are permitted since the compiler routine occupies less core storage. The length of integer variables is now ten digits. This allows integer arithmetic in the comparison of alphameric fields. Finally no object decks are produced, since C4D object programs are stored temporarily or permanently on the disk.

The class A subroutines are little changed from their PDQ ancestors. They do detect undefined symbols at run time and, in this event, return control to the executive routine. The relocatable subroutines have real or integer values according to their initial letters. They include all the original PDQ

relocatables, an absolute value routine for integers, a random number generator, and the MOVE routine[3] for disk seek, read, and write operations. The LOG subroutine is now called ALOG since it is a real function.

The executive routine determines the nature of each job from a control card. It then loads the appropriate system routine or object program from disk. It acts as librarian for programs on disk, it loads and relocates the relocatable subroutines, it controls dumps to disk or printer, and it prints headings and dates for each job. Finally it permits automatic exit from the system by loading another program not handled under the system.

Speed comparisons with other systems will only be possible after extended experience with C4D and will vary for different jobs and for different machine configurations. Unlike the IBM MONITOR system, C4D has no provision for assembly language programs and does not permit separate compilation of subprograms with local variable names and statement numbers. These limitations, however, must be balanced by important advantages for the installation which runs mainly FORTRAN programs. The system uses only 4% of the disk and the rest of the disk may be divided into program storage and working data storage, to suit the needs of each particular installation. The use of a second disk drive is readily enabled at system load time. In a two drive system, one drive contains only working data storage, and the other only systems and users programs. Other advantages of C4D are the efficiency of core utilization and the speed of disk operations for program segmentation and for data storage.

When C4D is submitted to the Users Group Library, the documentation will consist of three manuals for three distinct groups of readers. The Users Manual defines C4D as a programming language for users with a basic knowledge of FORTRAN. It describes those control cards of interest to the user and lists all error messages. The Operators Manual describes the control cards needed in running the system, the system sense switch options, loading the system, and interrupting the system for dumps. The Systems Manual is written for systems programmers who wish to correct, extend, or generally modify C4D itself, or who are just curious about how someone else constructs a system.

REFERENCES

1. F. H. Maskiell, "PDQ FORTRAN", U. G. Library Program 1620-02.0.031.

2. J. Snediker, C. T. Snyder, Jr., J. W. Burgeson, "FORTRAN II DIAGNOSTICIAN", U. G. Library Program 1620-01.6.019.

3. Mary Lynn McAteer, "MOVE", 1620 Users Group Newsletter, Volume 2, No. 5, p. 4, October 1964.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the contributions of many friends and colleagues.

John Burgeson of IBM whose FORTRAN II DIAGNOSTICIAN provided the original basis for the C4D diagnostic routine. Lawrence Powell of Pennsylvania Transformer for the dump-to-disk routine. George Moesta of Jones & Laughlin Steel Corporation for his printer-dump routine. James Taylor of Data Corporation for a faster routine for the compiler symbol table scan. Dick Pratt of Data Corporation for his preliminary version of 1443 AFIT SPS which was used exclusively in assembling the system. Manfred Fleiss of Jones & Laughlin and Joseph Smith of the RAND Corporation for contributions to the sample procedures. John Holmes of Cooper-Bessemer, Stewart Lee and James Field of the University of Toronto, and Donald Jardine of duPont of Canada, for their suggestions built into the original PDQ. Jack O'Keefe of American Bridge for the REREAD feature.

WISCONSIN ACCOUNTING LANGUAGE

Mr. Craig MacKenna
Computing Center
University of Wisconsin-Milwaukee

(SLIDE) WISAL, W-I-S-A-L, stands for Wisconsin's Accounting Language. WISAL is a key-word compiler, similar in this respect to COBOL. The source language involved is similar to current accounting language.

WISAL is natively a 1400 series system, the language having been devised and the first processor written at the University of Wisconsin Commerce Data Processing Center, by Mr. Richard W. McCoy and associates, for an IBM 1410. Potentialities are also inherent in the WISAL language for its use as an applications system--useful to business and industry as a system permitting their accountants to handle every day accounting transactions by computer with an absolute minimum of programming or computer familiarity. Emphasis in this talk, however, is on use of WISAL as an educational accounting system.

WISAL was first used at the University of Wisconsin in introductory accounting courses in spring of 1963. The University of Wisconsin-Milwaukee (hereafter referred to as UWM, while Madison as UW) first used WISAL in the fall of 1964. The fall 1964 project at UWM involved approximately 200 elementary accounting students. The fall 1964 project at UW involved about 300 elementary accounting students. (SLIDE) By fall of 1965, the UW Commerce

Data Processing Center intends to publish a book, dealing with the use of WISAL as an educational accounting system. It will deal mainly with the use of WISAL on the IBM 1410, but acknowledgement will be made of the fact that there is a 1620 WISAL. Since the book should deal only with source language considerations, it should be completely compatible with 1620 WISAL. Available from UWM will be a supplement to the book discussing things about 1620 WISAL which are different from 1410.

As previously mentioned, WISAL is natively a 1400 series system. Its very purpose and form of output are indicative of 1400 series influence; however, a joint University of Wisconsin-Madison and University of Wisconsin-Milwaukee venture into producing general 1620 WISAL compilers has been undertaken in the knowledge that many schools who will be interested in using WISAL as an educational accounting system have a 1620 at their disposal. This talk deals with UWM WISAL-D, a disk-oriented version of WISAL. We hope to have a non-disk version ready by fall of 1965. Naturally, the specifics will change in the non-disk version, but the source language should certainly remain the same. Either of these two versions will work on a basic 20-K machine; that is, no automatic floating point or other special features. The version with which we are specifically concerned today, that is the D version, the 1311 version, should work on either a Model I or Model II, under Monitor I or Monitor II; it can use a printer; it does not use index registers, but on a Model II they will be

turned off. We hope to have the 1311 version in the library by July of this year; the non-disk version in the library by September. 1620 source listings will be available from UWM, the library, and I believe I have some with me. Also there will be source decks and object decks for the processor and one problem. There will be a programmed solution for that one problem, the only one which has been written to date--The Wholesale Paper Company.

In a sense WISAL is a rigged system. Any problem to be run under it by students has to be completely worked out in advance; that is, it has to be handled by normal accounting means and the correct answers derived. This must be done because in WISAL there is the capability to produce for the accounting instructor at a later date a log of the errors made by his students running under the WISAL system. This error log is stored on disk in a permanently assigned area to be dumped by a separate program called WISLOG at such time as the instructor should desire a block of results or at such time that the permanently assigned area is filled. The instructor can do with these error lists as he sees fit; he can give them to his students as an aid in debugging or correcting their programs, or he can withhold them from the students and possibly use them as an aid in grading the student in his project.

Once the accounting problem has been worked out and coded in SPS in a format acceptable to the WISAL processor, it can be stored

on disk under Monitor like any other program or data block. At execution time WISAL can call any of several problems; that is, 1620 WISAL-D has multiple problem capabilities. It can handle up to four disk-stored problems under it at any given time. Input for the WISAL systems is via cards; output can be either card or printer under the control of a control record.

(SLIDE) This is a full WISAL output. It can be divided logically into five parts. Here, the first part, is a print-out of the source program. (SLIDE) This particular output is a simulation of that produced by a 1620 having a 1443 printer attached and printer output specified. With a card system, this phase of output is eliminated; there being no purpose in just duplicating the source program. However, even under a card system, this phase of output is represented by a punch-out of a copy of any card containing a source language error, followed by a diagnostic error message. (SLIDE) This second phase of output is a print-out of the reference section, as it is called. To each of the source language routines as written out above is attached a label. The student then sets up these labels in an array; the first field of the first line of this section being the label of that routine which the student wants to have handle transaction one. The second field on the first line is the label of that routine which the student wants to have handle transaction two, and so on through the complete set of transactions involved in the problem. Again, the outputting of this section is eliminated under a card output

system, but is still represented by diagnostic messages and punch-out of cards having errors--such errors having to do with the application of source language to particular transactions--a routine inappropriate to the transaction.

(SLIDE) This third phase of the output is that of a general ledger. This is output under either the card or printer system, that is, after the source program has been run in and executed. (SLIDE) Directly following this is the fourth phase, the accounts receivable trial balance. The fifth phase of output is, below here, an income statement: the figures here being derived from the particular accounts in the preceding phase. (SLIDE) The format of output in all phases and the assignment of particular amounts so as to derive this income statement and the following balance sheet is done completely under system control. (SLIDE) The sixth section is the balance sheet for this theoretical company and month--a page of assets and a page of liabilities. All the editing and formatting being done by the program in this printer version. (SLIDE) Most of the editing is still being done by the computer in the card version; however, with the card version, for this neat appearing type of output, the print-up should be subject to a 407 board split, the split being quite simple to set up. The last little line of the output --"BINGO"-- simply means that the student has "balanced the books".

You will notice that this comprises quite a few pages of output. On a card system a program having no source language errors

will punch about 275 cards, of which about forty will be blank. The criteria on cutting off the 275 card output is made variable.

(SLIDE) Now let us turn to the specific form of the source language. As mentioned before, this source language is similar to that used in current accounting practice. Here is one typical WISAL instruction, or typical WISAL sentence as it has come to be called: "DEBIT ACCOUNT 1111 BY AMT 1." This is the most basic one can get in a WISAL instruction, and no other instruction is really much more complex. All WISAL sentences start with a verb, DEBIT or CREDIT; an object, such as 1111 here--i.e. an account number, the account numbers being in the booklet which each student would have; and third, the AMT. The idea of using AMT 1 and AMT 2, etc. (they go up to AMT 9 in the 1620 WISAL system) to represent fixed fields in each transaction produced certain difficulties in the running of student programs. It had to be explained, either through the write-up or through individual questioning and answers, what each AMT represented in each transaction. Thus, certain rules were set up for "amounting". If a transaction for the month involved a cash amount, it was referred to as AMT 1; and if it involved a discount, it was referred to as AMT 2. (SLIDE) An alternate coding to solve some of this difficulty is now seen here: "DEBIT ACCOUNT 2113 BY CASHAMT" (CASHAMT referring to the amount of the cash payment or disbursement which is involved in the transaction). The three mnemonic AMT codings which have been developed are this CASHAMT, DISCAMT, the amount of a discount,

and VOUCAMT, the amount of a voucher. If there are ever amounts which cannot be classified strictly as cash, voucher, or discount involved in the transaction, you have to go back to the numeric system.

(SLIDE) This slide illustrates two flexibilities of the WISAL language. First of all, as in this line, you have the capability of saying CREDIT (CUST), that is, credit the account of a particular customer. The particular customer's account number, if any, involved in each transaction is stored with the transaction under the WISAL system, and thus the processor compiles instructions to pull this address, or account number, out of the transaction and then operate upon that. The rest of the sentence is the main illustration: (AMT 1 + AMT 2). This routine handles transactions involving a discount where it is desired to credit by the sum of the two amounts. Thus, we have the facility to do this all in one instruction. (SLIDE) Plus and minusing of amounts can also be done in words, as you see here: "DEBIT 1311 BY CASHAMT PLUS DISCAMT." One could also have the word MINUS as an alternative to the minus sign.

(SLIDE) A further illustration of amount flexibility: "CREDIT 4111 BY AMT OF THE CASH PAYMENT MINUS AMT 2." Here the idea of key-words comes into specific focus, aside from comments which could be to the front or to the rear of the sentence; that is, the idea of the key-words within the actual sentence. The key-words here are CREDIT, the four numerics in a row, 4111, and the key-words in the latter half, the amount portion: the word AMT, the

word CASH, (the words OF and PAYMENT being nonsignificant) and the word AMT 2. One thing not permitted is 2 AMT; that becomes a bit sticky.

You will notice here, and it has been brought to me as a criticism of this type of coding, that one variable or quantity is represented by a spread-out, noncontiguous group of symbols--AMT OF THE CASH PAYMENT, AMT and CASH going together to refer to one quantity. The criticism is that no other language does this, and that in an educational system you are trying to illustrate other languages. I think this type of coding is a nice feature, and if you agree with this criticism, there are always the CASHAMT and DISCAMT codings, which are handled the same way but which can be placed contiguously.

(SLIDE) This is the present extent of the WISAL language. Here is a general chart of the key structure of the WISAL sentence. There is the verb DEBIT or CREDIT, the object, four numerics representing a general account or the word (CUST) representing a particular customer account varying from transaction to transaction, and the various codings for amount--plus or minus AMT 1 through AMT 9, CASHAMT, DISCAMT, and VOUCAMT, or AMT..CASH, AMT..DISC, and AMT..VOUC. Possible future expansions include amortization subroutines, multiplication, literals--any features which could be of use in advanced accounting. But presently, since it is being used in an elementary accounting course, this is all that is needed.

This then is one WISAL sentence. A group of these may be put together to form a routine or subroutine. The key here is columns 8 through 11 which comprise a label. In processing, any card which does not have a label is assumed to be part of the previous routine: automatic continuation cards, so to speak, if you are thinking of FORTRAN. Or if it has the same label as the card or cards immediately preceding, it goes with the same routine. Only a different set of characters in columns 8 through 11 than those on the previous card having characters in columns 8 through 11 constitute a new routine. The routine may thus be as long as desired.

(SLIDE) The first part of the card, columns 1-6, is a page- and sequence number. These page and line number aid the computing installation and the student in correcting the program and prove invaluable in a project involving two to three hundred students.

Column 7 has traditionally been treated as comments indicator: if there is anything in that column, the card will be treated as a comment. This procedure has been expanded by several features which attempt to give the student a grasp of a flow of control through his program. If this was a WISAL coding form, you would see above the label columns, the words "if TRAN TYPE", and to the right of that, above the instructions, the word "THEN". (SLIDE) At the start of the student program, as you see here, there is a line with the word "START" in columns 7 through 11 and words "READ TRANSACTION" through the instruction section, followed by a card

"END (the letter E being in column 7) GO TO STATEMENT-PREPARATION". This gives the student sense of control: after all transactions have been processed, go to statement-preparation, go to preparing the output. (SLIDE) Then, at the end of the source program, there are two more lines. The first has the word "ERROR" starting in column 7; if error, then go to "CODE-NOT-FOUND", which is simply a diagnostic if you haven't matched the label appearing in the reference section with one in the source program. After that, the line "GO TO START", implying a loop and a flow of control through the program. (SLIDE) The concept that the student can work under is that execution starts at the top card--read a transaction. If all transactions have been processed, go to statement-preparation; if not, run through a sequence of "if tran type this then do that" --if tran type A5 do this, if tran type B7 do this, "if error go to code-not-found", and "go to start". This gives the student a concept of basic programming not otherwise found in a WISAL system without these features. However, since there is something in column 7 of each of these cards, they are actually treated as comments and are only used to give the student a sense of function greater than he would otherwise have.

The output can be produced upon having read the last card, the TITLE card. This card gives the student's name, section, name of instructor, and whatever else the student wishes to have punched thereon. Actually, execution is done as the reference cards are read in. Then, based on the number of source language errors and

reference section errors that were made by the student, a decision is made as to whether to punch out the rest of these cards. If a source program is particularly bad, there is no reason to continue on and punch out the full output. But, if the number of errors is less than that specified by the instructor, the computer continues to punch or print the full output.

(SLIDE) I have mentioned the possibility of instructor precoding transactions. This is done by control records. The control records are in this slide. There is a control record *PRINTER specifying that there is an on-line printer. A *PRECODE record sets the beginning total debits and total credits of any general ledger account or receivable subledger account to two specified values as to the right. To go with this is a *PRECODED record specifying that you are precoding particular transactions, as here you are precoding transactions 1, 7, 18, 22, 87, 88, 101, 102, and 124. The number of transactions in the Wholesale Paper Company program is 125. There is also one other control record, a *CRITERIA card, the four fields to the right representing the number of source language errors and reference section errors, of bad inclusion and omission, to be judged as the cut-off point for a full output.

(SLIDE) There is one more slide which we have here. This is a shot of the aforementioned error-log dump which the instructor can receive and do with as he pleases. You will notice in this error-log a copy of the title card, the number of the four types

of errors, and the magnitudes of the errors in total debits and total credits for each account which was different from that in the precoded problem. This then is the WISAL language and the 1620-WISAL-D processor. Thank you very much.

1620 SUPPORT FOR A DOCUMENT WRITING SYSTEM*

C. M. Thatcher

Pratt Institute,
Brooklyn, N. Y.

ABSTRACT

A primary use of the IBM Document Writing System is to prepare individually typed form letters from punched card input. Special characters are used to initiate tabulation, carriage return, shift to upper or lower case, etc. The insertion of these characters into the punched input for the letter itself is not difficult, but the punching of name-and-address cards requires concentration. A 1620 program which circumvents this problem is now in use. In addition to inserting all necessary control characters, the program expands a substantial number of abbreviations. As a consequence, name-and-address cards can be key-punched faster than the full name and address can be typed by hand, and can be used for both inside address and envelope address if desired.

The paper briefly describes the 870 System and its plugboard wiring for this application, indicates particular control symbols found to be most suitable, and considers the role of the 1620 in some detail.

* * * * *

* Paper presented before the 1620 Users Group Meeting in Miami, Florida, May 10, 1965.

The IBM 870 Document Writing System is essentially a data converter. The fully complemented system provides 18 different input-output combinations such as paper tape input to punch card output, punch card input to typewriter output, etc. The latter combination is particularly useful when a number of form letters must be individually typed: Punching the letter into a deck of cards makes it possible to produce as many copies as may be desired by the punch-card-to-typewriter combination--quickly, accurately, and with only intermittent operator attention.

Control over such typewriter functions as tabulation, carriage return, and case shift is exercised through control panel wiring. Each of the 12 special characters available via key punching (see Appendix 1) produces a pulse at a particular point on the control panel when it appears as an input character. This pulse may be routed by wiring to any desired typewriter control function or typewriter special character key, to obtain almost any desired output format.

For example, the control panel might be wired to make the typewriter carriage return whenever an "at" sign is encountered on an input card, while an equals sign might be used to cause the first letter of the following word to be capitalized. The need for an equals sign in the output document creates a problem in the latter event, but the typewriter keyboard and control symbols in use at Pratt Institute provide for a full range of output special characters, including the record mark.

A brief consideration of pertinent keyboard and control symbols should be of some interest and will be helpful when the part played by the 1620 is discussed in due course. First, note that the left and right parentheses appear twice in the keyboard listing in Appendix 2. This duplication led to the decision to use the left parenthesis to shift the typewriter to upper case and the right parenthesis to return it to lower case. Thus punching (7), for example, yields a question mark on the output document. Significantly, left and right parentheses can be typed by punching (9) and (0), respectively, making output parentheses fully available despite their input use as control symbols.

Similarly, the equals, at, and record mark signs can be typed by punching (+), (-), and (*), respectively. The corresponding input characters are thereby made available for control use, and the "at" sign is used for typewriter carriage return. The equals sign is used to shift the typewriter to upper case for the next input character only. This is a matter of convenience only, it being simpler to punch =A than (A) for the same result. The input record mark is used for various purposes which will be explained later.

The next step was to take advantage of four two-position latches on the 870 control panel. Depending on latch position, a given input character can be used to produce either of two different results. Latch position can also be controlled by punched special characters, and the left and right parentheses were selected for this purpose. This gives these symbols a dual function, but the two consequences are not incompatible.

Through the use of latch wiring, a punched (- causes typewriter tabulation (whereas - alone causes - to be typed); (= halts the operation pending operator intervention; and (/ ejects the card being read. A summary of the function of all special characters appears in Appendix 3. Note the use of the slash by itself for a second "convenience" function, namely a space followed by upper case for the next input character only.

A sample output letter is presented in Appendix 4, and the corresponding input card listing in Appendix 5. It can be seen that the special control symbols do not appear so frequently as to require extreme concentration during punching. Furthermore, it is possible to have a model copy of the letter typed at the same time that the input punch-card deck is being prepared. Any punching errors thus become immediately obvious and can be corrected on the spot.

The preparation of punch cards bearing name and address and a personalized salutation is something else again, as can be seen from the examples in Appendices 6 and 7. With control symbols appearing so frequently, error-free punching is difficult even with intense concentration, thereby making the punching of name-and-address cards a slow process. It was therefore decided to investigate the possibility of writing a 1620 program to take over the burden of inserting pertinent control symbols into symbol-free input cards. Initially, the program was intended to

(1) Replace any input double blank (used to signal the end of a typewritten line) with an "at" sign to return the typewriter carriage; and

(2) Insert an equals sign before any alphabetic character at the beginning of a line and replace any blank preceding an alphabetic character with a slash, thereby effecting capitalization.

Although these steps would greatly facilitate the punching of a "source deck", it quickly became apparent that an even more extensive use of the 1620 would be extremely advantageous. The program as originally envisioned was therefore successively expanded until it now provides for the following in addition to the two basic purposes cited above:

(3) An M, F, S, D, P, or R in column 1 followed by a blank in column 2 is automatically converted to Mr., Mrs., Miss, Dr., Prof., or Rev., respectively by the 1620.

(4) "and Mrs." can be inserted after any of the foregoing titles if desired, using program switch control.

(5) Source cards are punched with last name first, thereby facilitating proper alphabetic sorting and filing. The 1620 program reverses the order to obtain the desired output form.

(6) The 1620 generates a personalized salutation from the input last name and specified title, and adds this to the output object deck. Thus it need not be punched into the source card.

(7) A period is automatically inserted after any alphabetic character standing alone on the source card.

(8) A zero, 1, 2, etc., following the input name automatically generates Sr., Jr., II, III, etc., following the last name with proper punctuation.

(9) Cardinal numbers appearing as street names are converted to ordinal form; i.e., 18 Street becomes 18th Street.

(10) The comma separating city and state is inserted automatically. Zone numbers included on the source card are inserted ahead of the comma.

(11) Input cards in any single deck can contain up to 100 different two-character abbreviations which are expanded by the 1620 program. For example, ST can become STREET, PT can become PRATT INSTITUTE, BK5 can become BROOKLYN 5, NEW YORK, etc. The fact that the program permits changing the list of recognizable abbreviations is significant, since the most advantageous list may differ from letter to letter. Note that the particular abbreviations selected should not be letter combinations which might have to be used directly as such.

(12) Under program switch control, a common first line and/or common salutation can be inserted if desired. Thus DIRECTOR OF GUIDANCE need only be punched once if a number of letters are to go to guidance directors at several different high schools.

(13) One or more record marks are inserted into the output cards to permit the same cards to be used for typing inside address and salutation, for addressing envelopes, or for typing a proof list of names and addresses without operator intervention.

These features of the 1620 converter program greatly simplify the task of punching input name-and-address data, as can readily be seen from the sample input presented in Appendix 8. As a matter of fact, any reasonably competent key-punch operator can now punch the input card considerably faster than a good typist can type the full name and address by hand.

The use of the object deck produced by the 1620 program should be fairly obvious: The body of the letter is punched up separately and is

headed by a card bearing the date and accompanying punctuation and control symbols only. The name-address-salutation cards prepared by the 1620 are then inserted behind the date-line card, and the complete letter deck is ready to be run. Preparing one or two duplicate copies of the main letter deck makes it possible to change the name-address-salutation cards in one deck while the 870 is typing from another, with minimum loss of time.

One final feature of the entire program should be of interest. Since it was anticipated that letter writing would often be supervised by secretaries having no experience in plugboard wiring, it became desirable to use the same wiring for any of the three output uses envisioned--letter writing, envelope addressing, or name-and-address listing. This has been accomplished through the use of a single header card, into which a single code letter is punched to denote the operation desired.

Punching a P, for example, immediately ejects the punched header card and turns on the keyboard, card punch, and typewriter for succeeding cards. Thus the machine is automatically set for punching a new letter deck, with a model letter being typed at the same time. Punching a T, on the other hand, turns on the card reader and the typewriter for succeeding cards, yielding typewriter output from punch card input.

Punching an S or a ≠ also yields typewritten output from punch card input, but the former brings the operation to a halt when an input record mark is encountered while the latter ejects a card upon a record mark input. Study of the sample input card in Appendix 7 will show that the S punch accordingly stops the program after each name and address is typed, to permit changing envelopes in an addressing operation. The ≠ punch yields a proof list of names and addresses without pause, as can also be seen from the sample input card.

In summary, the 1620 converter program has made it feasible to use punch cards for typing inside addresses and salutations, addressing envelopes, and listing names and addresses. Without the program, the necessary key punching would be almost prohibitively painstaking and slow; with it, input punch cards can be prepared faster than the same information can be typed by hand.

It is likely that not very many Users will have an opportunity to put the program described herein to direct use. If the foregoing description encourages others to envision new uses for the 1620, however--uses not directly related to calculation--it has served its purpose.

APPENDIX

1. Special characters available via key punching:

   + - = * / ( ) . , @ $ ≠

2. Typewriter key combinations (in addition to usual upper and lower case alphabetic characters):

   ; # : " ¢ _ ? ' ( ) = @ ≠ ? % . , (
   1 2 3 4 5 6 7 8 9 0 + - * / $. ,)

3. Special character functions and type-outs:

Punched Character	Typewriter Response	Punched Character	Typewriter Response
(	Upper Case	=1 or (1)	Type ;
)	Lower Case	=2 or (2)	Type #
(-)	Tabulate	=3 or (3)	Type :
@	Return Carriage	=4 or (4)	Type "
		=5 or (5)	Type ¢
=P or (P)	Capital Letter P	=6 or (6)	Type _
/	Space and Capitalize	=7 or (7)	Type ?
		=8 or (8)	Type '
(=	Halt	=9 or (9)	Type (
(/	Eject card in process	=0 or (0)	Type)
≠	(Variable)	-	Type -
(@	Type /	= -	Type @
+	Type +	.	Type .
=+ or (+)	Type =	,	Type ,

4. Sample output letter (salutation name to be typed manually):

Dear Mr. :

Thank you for your letter of May 5. Coincidentally, I wrote you on the same date and you presumably already have my views on the matter.

Sincerely yours,

Otto McCanick
Director

OM:ibm

5. Corresponding punch card input (double line at right indicates continuation on same card to fill all 80 columns):

```
=DEAR/MR.(=
=3@@(-T)HANK YOU FOR YOUR LETTER OF/MAY 5. /COIN-@CIDENTALLY,/
                                                I WROTE YOU ON THE
bSAME DATE AND YOU@PRESUMABLY ALREADY HAVE MY VIEWS ON THE MAT
                                                TER.@@(--S)INCEREL
Y YOURS,@@@@(--O)TTO/MC=CANICK@(--D)IRECTOR@@(OM3)IBM(=
```

6. Sample output name-and-address, including personal salutation:

Mr. and Mrs. John J. Jones, Jr.
113 E. 45th Street
Bronx 18, New York

Dear Mr. and Mrs. Jones:

7. Corresponding punch card input:

```
@@=MR. AND/MRS./JOHN/J./JONES,/JR.@113/E. 45TH/STREET@=BRONX 1
                                                    8,/NEW/YORK@@+=DEA

))+R/MR. AND/MRS./JONES(3/
```

8. Corresponding punch card input to 1620 converter program:

```
M JONES JOHN J 1  113 E 45 ST  BX18
```

Use of the IBM Model I for the SIMPLIFICATION OF BOOLEAN ALGEBRA EXPRESSIONS

Charles A. Plesums

ABSTRACT

This program is an extension of the method developed by Quine and continued by Hoobert Huhta for the reduction to the simplest second order equivalent of a Boolean Algebra function of the sum-of-products form. In order to carry out the repeated comparisons involved in the process with maximum efficiency, the table-look-up hardware (addition hardware) of the 1620 Model I is used. As a result, a literal ouput can be obtained approximately ten times as fast as by previous 1620 computer methods.

This paper is concerned with a summary of both the definition of the problem and method of solution as evolved from the Quine method by Hoobert Huhta, and some of the unusual programming techniques used to implement the method. A complete description of the method and techniques can be obtained from the program of a similar name soon to be submitted for distribution by the User's Group.

In the logical design of digital circuits, as used in computers, control systems, and simulation, Boolean Algebra functions are derived from truth tables by means of the simple basic theorem. This form of the expression is usually not optimum for implementation; in other words, a simpler Boolean function can usually be found which performs the same job and is economically cheaper to construct.

In 1952 Quine developed a method of simplification that was quite simple and toutine; so simple that it is quite error prone. Basically, it involves the repeated application of the Boolean Identity, AZ+ AZ=A. where A is any Boolean expressinn. In the interest of conciseness, a review of the Quine method is omitted; such a description is readily available in many texts, such as Phister, LOGICAL DESIGN OF DIGITAL COMPUTERS, Wiley, 1958, p. 68.

It is not uncommon to represent Boolean minterms in a binary format. In the interest of conserving memory space in the decimally oriented 1620, this binary is converted to octal. In the octal format, the condition of three variables can be represented in a single memory location, and the identity of each of the bariables is maintained. However, in the binary or octal form a missing variable is not distinguishable. The Huhta method, therefore, generates a second number, called the set identification number, which represents, in octal form, the eliminated variables. Therefore, a single term in the Huhta process is represented by two octal numbers; the set identification number stating which variables are missing, and the term representing which variables are present in the true (uncomplemented) form.

Subsequent comparisons in the Quine process need only take place

among those terms that have the same variables. In the Huhta method comparison is executed only within a set of terms with the same set identification number.

In the use of the Quine method, the same term often evolves from entirely different combination of terms. In the Huhta method the duplicated terms are eliminated from processing by the following procedure. In the comparison process the octal weight of the eliminated variable is obtained. If this is greater than the value of the set identification number (SIN) of the terms that were being compared, the process continues like Quine. If the weight of the eliminated variables is less than the SIN, the simplified term will be one of a duplicated pair, and is not carried to the next level of simplificatinn.

In the comparison process, we say the terms combine if they differ in the state of only one variable; in other words if they differ in one binary place. To discover this condition, we must perform a digit-by-digit table look-up. From this look-up, three conditions can evolve: 1) the digits are the same, so the variables are the same, and the look-up may proceed. 2) the octal digits differ in more than one binary place, and therefore the terms are not combinable. 3) the digits differ in only on binary place, and the table must yield the octal weight of the place that differs. Note that the terms are cominable only if condition three exists in one and only one octal digit, and condition 2 never exists.

To perform this table look-up, the addition tables are replaced with a combinability table and the terms are added. Condition 1) produces a zero as a result of the "addition," condition 2) produces a record mark, and 3) produces the weight of the different variable. The "sum" is scanned for record marks or more than one digit, which indicate non-combinability. The table look-up and scan, including the modification of the table area, is executed in 7 to 11 instructions, depending on the data.

Another major programming procedure is the use of the process that has come to be known as the shift. There are several tables of numbers of which only one element is dealt with at a time, and after it is used, it is never referred to in that array again. Rather than deal with the relatively major programming effort of address arithmetic and instruction modification, as well as counters to indicate the extent of the array, only the first element of the array is dealt with. After finishing with the element, a Transmit Record is executed which removes the first element and shifts all the following elements one element closer to the beginning of the array. In addition, the presence of the record mark at the end of the array and repeated in each unused element by the process is used as an indicator of the end of the array so that no count must be maintained of the number of elements present.

The present form of the program requires a minimum 1620 model I with card I/O, and can handle problems of 15 variables, 150 minterms, 650 or 800 intermediate terms, and redundant minterms. It produces a literal output and a punched output for use by subsequent programs which have not yet been developed in SPS. The write-up of the program soon to be distributed by the User's Group contains about twenty pages describing the Huhta Method in detail; those interested should obtain a copy.

Further development in the area of programming Boolean Algebra can follow two paths...There is a need for a program to select the essential terms and non-essential prime implicants to complete the solution, and a whole new area of research would be the development of a method of simplification to other criterion, such as the cheapest combination of pre-packaged circuits; the fewest number of interconnections, or some other criterion.

AN APPLICATION - PLOT A BLOCK DIAGRAM

by

Theodore M. Hartz and Susannah H. Young [1/]

[1/] Members of the Data Processing and Mathematics Section, Radiation Surveillance Center, Division of Radiological Health, Bureau of State Services, Public Health Service, Department of Health, Education & Welfare, Rockville, Maryland.

Presented at the Eastern Region 1620 Users Group Meeting
May 9-11, 1965, Miami Beach, Florida

INTRODUCTION

The function of a block diagram, as the term is used here, is threefold:

(1) To provide the programmer with a means of visualizing, during the developmental stages of programming, the sequence in which logical and arithmetic operations should occur and the relationship of one portion of a program to another;

(2) To become a programmer's tool, serving as his guide and check-off sheet; and finally

(3) To go into the volume marked "DOCUMENTATION" and there to serve as the record used for future modifications and/or reference.

The block diagram used in stages (1) and (2) can be rough or smooth, and it can have one or one hundred revisions squeezed in or added on while the program is being developed, written and debugged. But for stage (3), a fine, smooth, complete, and up-to-date block diagram must be produced at documentation time; and this is always a tedious job for the programmer.

The program here presented is an outgrowth of stage (3) requirements because the redrafting of detailed block diagrams for an SPS program of over 4,000 instructions had been a most time-consuming chore for the impatient programmers. After this job had been completed, thoughts turned to the possibility of having the plotter assist with such labors in the future.

A short program was written to illustrate the various geometric figures proposed, showing assorted sizes and proportions along with suggested layouts for printing instructions. Execution on the plotter was a great success; and, after selecting the sizes and scales to be used, work on this program began.

Before proceeding further to describe this program, it would be well to divert for a minute to explain that the geometric figures selected are those in use in our office, and for the purposes stated. They are described in a "Reference Manual" which was distributed in May, 1963, to personnel in the Data Processing and Computations Section of the Division of Radiological Health.

Three geometric figures are used here and they are:

(1) Rectangle - to represent operation,

(2) Diamond - to represent decision, and

(3) Circle - to carry label and to represent branch.

PROGRAM

A description of each of these four parts now follows:

PART I -- ANALYZER:

(A) Initializes plotter, (sets up scales and moves pen to start), sets up margins, and draws a horizontal line to mark the top of a page.

(B) Initializes library (OPCODE) and corresponding table of geometric figures required (ROUTIN).

(C) Clears out input area, inputs card in alphameric format and sets up OPCODE field.

(D) Searches through library for required OPCODE and branches to the appropriate ROUTIN or to ERROR.

(E) Tests for amount of writing space still available on plotter paper and goes to new line or new page if required.

When it is determined by the analyzer that a new line or new page is needed, this routine will cause the plotter to:

(1) Draw a 0.8" diameter circle to show a branch from the mainline program,

(2) Write inside the circle the word "NEXT LINE" or "NEXT PAGE" as the case may be, and

(3) Initialize at either the new line values or at the new page values. In the latter case, a horizontal line is drawn to mark the start of a fresh page.

PART II -- LOCATER

After the opcode has selected the proper geometric figure and after the plotter has drawn that figure, the locater is called on to find the operands required. If the locater finds the items required, it will cause the plotter to draw them; otherwise, there will be a branch to ERROR.

PART III -- ERROR

This routine outputs error messages on the typewriter, if the other routines have indicated there was an error.

PLANNING AND PREPARATIONS

Each of these geometric figures was laid out to scale. Provision was made in the internal layout for sufficient printing to accommodate, in its entirety, each instruction being plotted. Space was also provided for a label, if present, in an appended label circle. Arrows needed to link the figures and to show directions of flow were laid out to scale. Subroutines were written to draw each one of them using the CalComp Plotter. Subroutines were also written which would locate:

(1) P operand of up to 6 characters,

(2) P operand of up to 10 characters,

(3) Q operand of up to 10 characters, and

(4) Flag operand.

Margins and layouts were established, and it was decided to use an 8 1/2" x 11" paper format (as plotter paper is 11+" wide) with 1 1/2" side margins. A subroutine was written to compute line and page consumption, and to draw a horizontal line every 8 1/2" to show where to cut the paper roll into separate sheets. The last preliminary task was to list all the opcodes which an SPS program could contain and alongside each opcode to write an English-language translation of an instruction using that opcode.

These English-language translations contain the key prepositions which differentiate the various operand functions and which, therefore, sort the opcodes into natural groups. This is shown in the following examples which concern flag operations. The first inclination in grouping opcodes would be that SF, MF, and CF would probably be handled as one natural group, but now consider their English-language translations and the key prepositions contained in each.

CF INPUT means Clear Flag at INPUT

MF INPUT, B means Move Flag from B to INPUT

SF INPUT means Set Flag at INPUT

The odd one here is MF as it uses 2 prepositions and has 2 operands, while CF and SF each use one operand and use the same preposition.

These natural groups of opcodes were further regrouped into a larger category by geometric figures required:

Operation, Branch with Decision, Special, and Branch without Decision.

A program outline developed naturally and was in four principal parts, as follows: Part I -- Analyzer; Part II -- Locater; Part III -- Error; Part IV -- Plotter.

PART IV -- PLOTTER

PLOTTER is the last major portion of this program and consists of seven separate parts. It is in this portion where the figures are plotted and the instructions are written.

A. Branch

Branch functions were the first to be defined and plotted. These were done by first grouping together similar kinds of branch opcodes after the pattern of their English-language translation. From this grouping came two subdivisions:

Branch-without-a-decision, and

Branch-with-a-decision

1. Branch-Without-a-Decision

To represent this kind of instruction in a block diagram required only a circle, (the convention used by this office for "Branch"); and this full circle, (0.8" diameter), had already been laid out and plotted above as part of the ANALYZER. No connection to the next instruction was needed for these opcodes, (such as BB), so the only task remaining was to insert the instruction in the circle. In all but two cases, this meant to write only the opcode. In these last two cases, (B and B_7), the P operand is important, while the OPCODE, Branch, can be implied by the circle. Therefore, a 6-space area was provided across the horizontal diameter of the circle in which to write a P operand. No connection to the next instruction was needed in this case either. For all these opcodes, the plotter was moved in the pen-up position to the starting point of the next figure.

2. Branch-With-a-Decision

This is the second branch subdivision and includes four opcodes for branch depending-on-conditions-in-the-Q-operand, plus the 26 opcodes in the BRanch-on-INDicator list. By converting an instruction using one of these opcodes into an English-language equivalent, their block diagram format was developed naturally, thus:

Opcode	P operand	Q operand	
BE	ALPHA	NONE	means BRANCH, if Equal/Zero Indicator is ON, to ALPHA
BNF	ALPHA	BETA	means BRANCH, if NO Flag in BETA, to ALPHA

It became evident from these translations that all factors involving the decision were self-contained in an OPCODE which has no more than four characters, yet can have a meaning as involved as "Branch if Sense Switch 1 is NOT on." On the proposed block diagram, all these factors would be represented by a diamond (decision) and a circle (branch) with arrows as needed and a part-circle for the label, if present.

A diamond was laid out with dimensions of 7 x 10 units which would reduce to plotter scale 0.7" x 1.0", very close to template size and proportions. To provide for the possible presence of a 6-digit label, a second circle (a part-circle with an 0.8" diameter) was laid out to attach to the diamond. The branch circle had already been laid out and plotted above as part of the ANALYZER. Arrows were needed to link the geometric figures and to show the direction of flow, and these were laid out to have an overall length of 0.3". The smallest print size available in the annotation routine (at the vertical angle) was used throughout the entire program.

The last task remaining was to plot (to write in) the instruction, but because these instructions differ in format, their layouts in this block diagram had to differ in the same way.

The 26 BRanch-on-INDicator opcodes were very direct and were no problem to lay out. Across the inside length of the diamond, a 4-space area was provided in which to write a 4-character (maximum) OPCODE. And similarly, across the horizontal diameter of the circle, a 6-space area was provided in which to write a maximum 6-character P operand.

The four BRanch-on-Q instructions were written out on 2 horizontal lines inside the diamond with the opcode and the word "IN" on the first line, and the Q operand on the second line -- the whole being read as "Branch NO Flag in Q" and, then, the entire P operand was written across the horizontal diameter of the circle, showing the Where of the Branch.

B. Operation

Operation functions were the next group to be defined and plotted. Grouping similar kinds of opcodes together had provided the groups of naturally similar branch instructions. The same system was continued for operation functions, and provided, not two, but eleven naturally-similar groups. Further study of these operation groups showed that they would all utilize the same operation box, and the same appended label-circle, and the same connecting arrow; but that they would differ in the English-language translation of the instruction.

It was possible, therefore, to have one common routine for all these opcodes up to the annotation of the instructions, at which point each of these groups would be handled according to its own unique requirements. These eleven operation groups are:

Subroutine Name	# of Opcodes	List of Opcodes
FQTP	9	TF, TFM, TR, TD, TDM, TNF, LD, LDM, MF
ADCOM	4	A, AM, CM, C
SUBTR	2	S, SM
MULDIV	4	M, MM, D, DM
STRIP	1	TNS
READ	8	RN, RNTY, RNPT, RNCD, RA, RATY, RAPT, RACD
WRITE	12	WN, WNTY WNPT, WNCD, DN, DNTY, DNPT, DNCD, WA, WATY, WAPT, WACD
SUPER	16	SK, RDGN, WDGN, CDGN, RTGN, WTGN, CTGN, RDN, WDN, CDN, RTN, WTN, CTN, SEEK, GET, PUT
FLAG	2	SF, CF
CALL	1	CALL
DRAW	1	DRAW

The operation box was laid out with dimensions of 8 x 11 which would reduce to plotter scale 0.8 x 1.1 inch, very similar to template size and proportions. The length was sufficient to accommodate 10-characters of writing, and the width sufficient for five lines of writing. To provide for the possible presence of a 6-digit label, a third circle (a part-circle with an 0.8" diameter) was laid out to attach to the operations box at its upper left corner.

The instructions were laid out as follows:

FQTP	Line 1.	OPCODE FROM
	2.	Q operand
	3.	TO
	4.	P operand
	5.	Flag operand
ADCOM	1.	OPCODE
	2.	Q operand
	3.	TO
	4.	P operand
	5.	Flag operand

SUBTR	Line 1.	OPCODE
	2.	Q operand
	3.	FROM
	4.	P operand
	5.	Flag operand
MULDIV	1.	OPCODE
	2.	P operand
	3.	BY
	4.	Q operand
	5.	Flag operand
STRIP	1.	OPCODE FROM
	2.	P operand
	3.	TO
	4.	Q operand
	5.	Flag operand
READ	1.	OPCODE INTO
	2.	P operand
	3.	---
	4.	---
	5.	Flag operand
WRITE	1.	OPCODE FROM
	2.	P operand
	3.	---
	4.	---
	5.	Flag operand
SUPER	1.	OPCODE PER
	2.	P operand
	3.	---
	4.	---
	5.	Flag operand
FLAG	1.	OPCODE AT
	2.	P operand
	3.	---
	4.	---
	5.	Flag operand
CALL	1.	OPCODE
	2.	P operand
	3.	---
	4.	Q operand or Blanks w/MES3
	5.	---
DRAW	1.	OPCODE
	2.	---
	3.	---
	4.	Fourth operand
	5.	---

C. Branch and Transmit (BRANT)

Another opcode represented by this same operation box and label circle is the one for BT and BTM -- here called BRANT. Because of its threefold function, it follows a unique format and has to be so treated. This unique instruction format is due to a two-way flow between the operation box and the branch circle, and this is shown by the use to TWO arrows, (one down and the other up). The layout of the instruction further conveys the threefold meaning of it -- as follows:

The Branch circle contains the P operand to which the program branches and from which it later returns. The OPCODE is on line (1) of the operation box and the Q operand, the address of the data transmitted to P-1, is on line (2). Flag operands if present are on line (5), while line (3) and (4) remain blank.

D. Miscellaneous (MISC)

Another operation function which is somewhat similar to the previous eleven is MISC. This routine concerns the control operations which often appear in profusion in a program but which one may wish to omit from a block diagram. Except for "K", these opcodes have a complete meaning in themselves -- are self-contained, and require no operands. To accomodate this kind of opcode, a switch setting is provided here, and this routine will then perform the following:

Search for a label and, if one is present, draw it with its label circle, proceeding right along to draw the operation box and then the opcode. Then the opcode is tested and if it is "K", a further search is made for operands and those present will be drawn, along with an arrow to the next instruction. In case the OPCODE is not "K", an arrow is drawn to the next instruction.

If, however, the first search finds that this instruction does NOT have a label, the program then branches to find how SS1 is set. If SS1 is ON, the program returns to draw the operation box, the opcode and the arrow, as usual; but if SS1 is OFF (and there was NO label) this causes the entire instruction to be omitted and the program returns to START and the next instruction.

E. Plot

A separate treatment for the CalComp macro-instruction PLOT was required because of the 2 sets of coordinates and their variations in length. To accommodate the different lengths of the sets of coordinates, two sizes of operation boxes were laid out, (0.80" x 1.10" and 0.80" x 2.20"), and made available to accommodate these instructions. This routine decides which size to use after:

(1) searching out the first pair of coordinates and counting their combined number of digits, (2) searching out the second pair of

coordinates and counting their combined number of digits, and (3) comparing each total to 10. If both are 10 or less, the small box is used, otherwise, the large one is used. A label circle is also provided, if needed, and the instruction is written out in the operation box in this format:

Line (1) OPCODE FROM

Line (2) Coordinates (1) and (2)

Line (3) TO

Line (4) Coordinates (3) and (4)

Line (5) ---

F. Declarative Operations (DECLOP)

The last group to be described is a 21-item list of declaratives:

DS, DSS, DAS, DC, DSC, DVLC, DAC, DSAC, DSA, DSB, DNB, DDA, DGM, DTN, DTA, DCN, DCA, DPTN, DPTA, and DDW.

These are not a part of a block diagram, but still one may not wish to omit them entirely. To accommodate this kind of situation, a switch setting is provided here, and this routine will perform as follows:

If SS2 is ON, the program will punch a card for each item in this list. But if SS2 is OFF, the program will omit the declarative completely and return to START and the next instruction.

G. Special

In this last group, the opcodes DEND and DORG were considered as a special class. The program will perform as follows:

DORG -- Ignore this opcode and proceed to the next instruction.

DEND -- Call EXIT, thus bringing an end to the plotting.

CONCLUSION

Machine requirements for this program are:

1620 with 40K memory, Monitor System (one disk drive), card reader/ punch, CALCOMP on-line plotter, and MF-TNF-TNS-automatic divide.

The program here presented is operational and has already proved to be extremely valuable. One of its hidden assets is its flexibility. Changes can be very easily introduced, and several have already been discussed. These include such additions as:

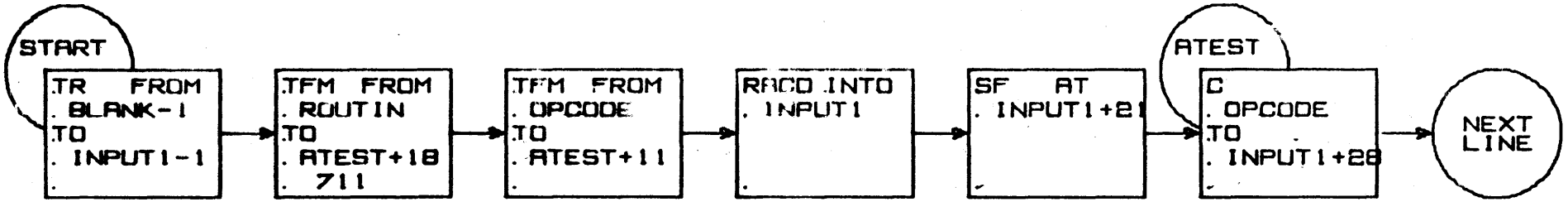
(1) A fourth geometric figure to contain comments and remarks,

(2) A page numbering feature, and

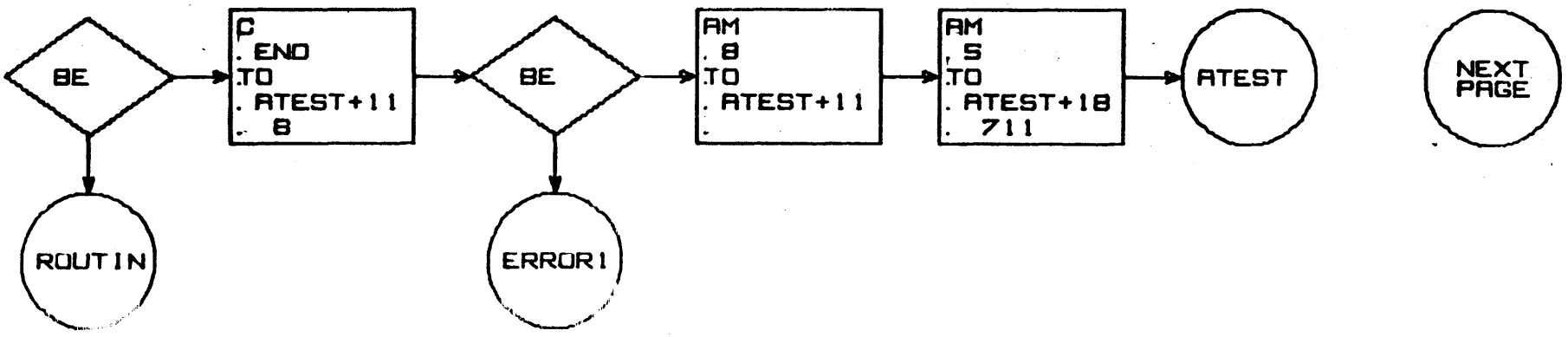
(3) Other alterations as may be required to meet a user's individual requirements.

A representative assortment of SPS Source-Program instructions together with their block diagram format, as produced by this program, is attached (see Appendix A) to illustrate typical plotter output.

```
1066START TR   INPUT1-1  ,BLANK-1     .    ,
1070      TFM  ATEST+18  ,ROUTIN     ,711,
1080      TFM  ATEST+11  ,OPCODE     .    .
1090      RACD INPUT1    .           .    .
1100      SF   INPUT1+21 ,           .    .
1110ATEST C    INPUT1+28 ,OPCODE     .    .
```




```
1120      BE   ROUTIN    ,          ,6   ,
1130      C    ATEST+11  ,END       ,6   ,
1140      BE   ERROR1    ,          ,    ,
1150      AM   ATEST+11  ,8         ,    ,
1160      AM   ATEST+18  ,5         ,711,
1170      B7   ATEST     ,          ,    ,
```



ASSEMBLING SPS I WITH MONITOR I

T. I. Markland
IBM Systems Manufacturing Division
Poughkeepsie, New York

The floating point formats of SPS I and SPS II or SPS II-D (Monitor I) pose the greatest problem to program conversion because of their incompatibility. SPS I uses Excess 50 floating point notation while SPS II-D does not. The most significant difference between the two systems of notation is that SPS I uses only one field while SPS II-D uses separate fields for the mantissa and the characteristic. Conversion from SPS I to SPS II-D requires changing the format of each number defined in the program, and every reference to these numbers; e.g., each Transmit Field instruction, used to transmit a floating point number, must be changed to a TFLS instruction. This problem becomes troublesome if the program involved is one of a set of programs and the output from one is the input to another; i.e., the AUTOSPOT System.

Because the operation codes for both floating point formats are the same, it is not possible to write a separate set of subroutines for SPS II-D without modifying both the system op-code table and every floating point op-code in the source deck.

The IBM 1620 Monitor I System Reference Manual (Form C26-5739-3) explains a method of adding user-written subroutines to the Monitor System. The application of this method to the conversion of the standard subroutine set will be the main topic of this paper.

Since the SPS I subroutines did not have the advantage of variable length mantissa, only the fixed length subroutine set in SPS II needed modification. Although the fixed length subroutine set is much simpler than the variable length set, the task of thoroughly reviewing and understanding these subroutines is not particularly pleasant. Besides, most programmers do not care how the floating arctangent subroutine works as long as it works.

One of the requirements of placing a user-written subroutine on the file is a knowledge of the PICK subroutine. The PICK subroutine is used by every other subroutine, with the exception of fixed point divide, to set up the addresses and data for the subroutine. The PICK routine can be modified without much involvement in the logic of the other subroutines. Only slight modifications are necessary in the other subroutines to make them compatible with the new PICK subroutine.

Knowledge of the rules and methods of relocation for subroutines is also important. First of all, each subroutine is a relocatable program. Regardless of the main program location in storage, PICK and the other subroutines will follow. Therefore, certain addresses and data areas used in the subroutines will be located in a different area depending on the last location used by the main program. Also, each reference to these areas must address the correct location; hence, many instructions in the subroutines must be corrected during the loading of the subroutine. Furthermore, the arithmetic and functional subroutines use areas that are defined in PICK. This requires that the arithmetic and functional subroutines be modified for their own location in core storage and the location of the PICK subroutine.

One area which requires consideration is PCK. This is defined in the PICK subroutine at the actual address 02365. This area is used for communication between the main-line program and all of the subroutines. The address of the entry point to PICK is always located at PCK. The addresses of the entry points to the other subroutines are located in five-digit fields below PCK and all of these addresses are loaded into this area when the subroutines are loaded into core storage. Addresses above PCK are used by PICK and the other subroutines to communicate between themselves and the main line program. A review of the method of communication shows that:

The first instruction of the linkage is TFM PCK + 10, * + 19. This instruction transmits the address of the high order digit of the address of A operand to the common communication area. The second instruction is B7 -(PCK - 5*X). This instruction causes the computer to execute the first instruction at the entry point to the called subroutine (FA, FS, etc.). The branch is indirect to the common communication area which was set to the proper address when the subroutine was loaded. X represents the number of the entry point in the subroutine entry point list, each entry point having its own address area. After the branch, a DSA is effectively assembled with the addresses of the two operands.

The subroutine now executes a TFM to set up a return to itself and branches to PICK. PICK then executes a TR, with an indirect Q address of PCK + 10. to bring the address of the operands to the common area; adds 11 to the address at PCK + 10 to find the main line return address; places the B operand in an area referred to as BETA; and, returns to the arithmetic or functional subroutine. The subroutine calculates the result, places it in an area labeled ALPHA, and returns control to PICK. Finally PICK handles all the error codes and arithmetic indicators, places ALPHA in the A operand in the main program, and branches back to the main program to complete the cycle. This is the general logic but some subroutines do not take the complete advantage of all the functions of PICK.

Since PICK uses the PCK addresses to obtain the operands, sends the operands to the appropriate subroutine, and returns the result to the main program, it may be possible to modify PICK, to convert a floating point number which is in the Excess 50 notation, to SPS II-D notation and then reconvert the results before returning them to the main program. And if this can be done without changing the results of assembling an SPS II-D program, it may be possible to assemble source decks for either SPS system.

One slight problem which may necessitate changes to the SPS I source deck is created by the main program using address modification. Changing the addresses of the operands in the linkage, or branching around linkage will cause problems. SPS I and SPS II-D assemble linkages of different lengths and with the addresses of the operands in different locations. A short review of all references to addresses near a macro instruction should solve this problem.

Some of the problems encountered during the modification of the SPS II-D subroutine set 01 were:

a) SPS I subroutines place a product or quotient at location 99 rather than in the A operand area.

b) The arithmetic subroutines set up the A operand and expect to find them in the SPS II-D format (this could require another SPS I to SPS II-D conversion routine in each of the arithmetic subroutines).

c) Some of the subroutines operate on the operands directly (rather than on ALPHA and BETA) by indirectly addressing to those addresses in the common PCK area.

d) An SPS II-D generated result may not be converted correctly back to SPS I due to overflow or underflow.

e) SPS I and SPS II-D error recognition techniques are different.

f) Certain constants and data areas are defined in PICK and referenced in the other subroutines.

g) The SPS II-D floating shift operations do not address low order digits of floating point field.

A suggested solution for each of the above problems will follow as the logic behind the general technique of conversion is reviewed. The following steps will explain the operation of PICK and how the conversion is accomplished in the PICK subroutine:

1. Reset error digit 00401.
2. Move addresses of operands from main program to PCK area.
3. Calculate main line return address.
4. Calculate addresses of mantissa and characteristics of both operands.
5. Reset a switch used to determine the floating point format when converting back (CBSW).
6. Proceed to the SPS II-D set-up routine if either an FSLS or FSRS operation has been called for, or the floating point operand is in SPS II-D format. If none of these conditions exist, continue to the SPS I to SPS II-D conversion routine.

NOTE

A switch must be set in the FSLS and FSRS subroutines for this test because of problem (g) above. The switch may be set upon entering either routine and reset upon returning from PICK. The format check is made by subtracting one from the address of the B operand and checking for a flag on that digit. The following steps follow the logic of the SPS I to SPS II-D conversion.

7. Set CBSW and reset multiply-divide switch (MDSW).

NOTE

The MDSW is set in either the FM or FD subroutines. This switch takes care of problem (a).

8. Move B operand to BETA-2.

9. Check for digit at BETA-9. If no digit is present, place $\overline{0}000000\overline{99}$ in BETA and proceed to step (11).

10. Convert BETA to SPS II format by subtracting 50 from BETA-10 and moving the result to BETA.

11. Skip ALPHA set-up if not arithmetic subroutine.

NOTE

This test is accomplished by checking for the presence of a flag in the PCK common area. The flag is always placed by each subroutine and then is removed by each non-arithmetic routine. This satisfies the requirement of problem (b).

12. Convert ALPHA to SPS II as per steps (9) and (10).

13. Return to correct subroutine.

NOTE

If the format was SPS II-D, PICK places the B operand in BETA and checks for an arithmetic subroutine. IF FA, FS, FM, or FD has been called, ALPHA also will be converted.

It will be assumed at this point that the functional or arithmetic subroutine has been executed, the result of the operation has been placed in ALPHA, and control has been transferred back to PICK. Any modifications required in the arithmetic and functional subroutines will be covered later. The remaining steps complete the explanation of the PICK subroutine.

14. Set error digit at location 00401 if required.

15. Return ALPHA to A operand and return to main program if CBSW is reset.

NOTE

If CBSW is set, the original operands were in SPS I format.

16. Proceed to step (19) if there is a digit in the high order position of the ALPHA mantissa.

17. Set ALPHA to all zeros and proceed to step (22) if the error digit at location 00401 is not a record mark (overflow or underflow).

18. Set ALPHA TO $\bar{5}000000000$ and proceed to step (22) if the error digit is a record mark.

19. Add 50 to the characteristic of ALPHA.

20. Place all zeros or all nines in ALPHA if the result of the add is either negative or overflows (problem d above).

NOTE

If an underflow or overflow occurs as a result of the operation or the conversion, an error routine places the zeros or nines in ALPHA, types out the main line return address and the addresses of the operands, and halts (problem e); depressing START will cause PICK to continue as if the error had not occurred.

21. Move ALPHA to ALPHA-10 (return to SPS I format).

22. Set HP and EZ indicators by adding zero to the result.

23. Place the result in either the A operand or location 99 depending on the MDSW.

24. Return to main program.

This completes the part of the procedure which is handled by the PICK subroutine. The remainder of this paper treats the conversion of the other subroutines.

The first step is to visually scan the listings of each subroutine to find all references to PCK + 15, PCK + 20, PCK + 25, and PCK + 30. These are the addresses of the characteristic and mantissa addresses for the A and B operands. Since these operands may be in SPS I format, they should not be worked on directly. However, the numbers in the PICK subroutine already have been converted and placed in the correct format in ALPHA and BETA.

The subroutines may work on ALPHA and BETA directly by changing

PCK + 15 indirect to ALPHA direct;
PCK + 20 indirect to BETA direct;
PCK + 25 indirect to ALPHA - 2 direct; and,
PCK + 30 indirect to BETA - 2 direct.

NOTE

These changes are not required in FSLS, FSRS, TFLS, and BTFS because they are not included in the SPS I subroutine set.

The last problem (f) is caused by certain data areas being defined in PICK and referenced in the other subroutines (ALPHA, BETA, etc.). First, it is necessary to determine which areas are referenced in the other subroutines. One method of doing this is to assemble all of the subroutines with the exception of PICK and list all errors indicating undefined symbols. The following labels are those requiring definition:

ALPHA	LCN1	NOSDIG	SIGN
AZERO	LCN2	ONEZ	STORE
BETA	LCN3	OVFL	UNFL
CZERO	LCN5	PCK	ZRES
FAC	LOGE	SAVE	
FLONE	MDSW*	SFTS*	

The Monitor System symbol table provides an easy way of allowing symbols defined in one program to be used in another program. Symbols may be defined in this table by using the SPSLIB portion of Monitor; however, it is necessary to determine the relocation address of each symbol. To do this, it is necessary to assemble PICK (relocatable) and note the addresses of each symbol listed above.

The control cards required to define the System Symbol Table are as follows:

```
≠≠ JOB 5
≠≠ XEQ SPSLIB
* DEFINE SYSTEM SYMBOL TABLE
          ALPHA      -00066
          AZERO      -01134
etc.
* ENDLIB
≠≠≠≠
```

The symbol in each detail symbol card starts in column 6 while the address starts in column 16. Addresses which are to be relocatable should be preceded by a minus (-) sign. NOSDIG and PCK are the only addresses which are not relocated.

Only half of the problem has been solved; it is necessary to identify for the subroutine supervisor program, which loads all our subroutines, those addresses which should be relocated according to the actual location of the subroutine being loaded, and those to be relocated according to the actual location of PICK. This is accomplished by defining a "pseudo constant (DC statement)" in the source deck. An explanation of the use of the pseudo constant is given in the Monitor Manual under the heading "Operands that are a Function of Pick and/or Mantissa Length." The only "modifiers" which are required are 0 and 5, since the fixed length subroutines are not affected by mantissa length. A check of each subroutine, after all other changes have been made, will indicate what must be done to the pseudo constants to make the subroutines function correctly.

NOTE

PCK is not relocated and does not depend on the location of PICK.

*These were defined specifically for the modified subroutine set.

The other changes required in the arithmetic and functional subroutines include:

1. Changing all direct returns to the main program (branches to PCK + 10 indirect), to branches to the second portion of PICK.

   NOTE

   This should not be done for the FSLS, FSRS, TFLS, and BTFS subroutines.

2. Set switch in functional subroutines and TFLS and BTFS to stop ALPHA set-up (this may be done by clearing the flag which is always set on the highest order digit of the address of the return to the functional subroutine (PCK + 1).

3. The last change requires setting and resetting a switch in the two-shift subroutines (FSRS and FSLS). This switch (SFTS) also used to inhibit ALPHA set-up is to be set immediately upon entry to the subroutine and reset immediately upon returning from PICK.

Before loading any of the subroutines onto the file it is first necessary to delete them in their existing form. This is done with the following control cards:

```
≠≠ JOB 5
≠≠ DUP 5
* DELET          0100
≠≠≠≠
```

The numerical portion of the *DELET card is the DIM Entry Number of the PICK subroutine for subroutine set 01. The DIM Entry Number for the proper subroutine is entered in columns 13-16. The DIM Entry Numbers for each subroutine in set 01 are:

PICK	0100
FA and FS	0102
FM	0104
FD	0105
FSQR	0106
FSIN and FCOS	0107
FATN	0109
FEX and FEXT	0110
FLOG and FLN	0112
FSRS	0114
FSLS	0115
TFLS	0116
BTFS	0117

NOTE

The double subroutines are really one subroutine with two entry points. Since the fixed point divide routine simply uses the automatic divide instruction in this subroutine set, and there are no references to the PICK subroutine, it is not necessary to change this routine at all.

The last thing to be done is assemble each subroutine and store it permanently on the file. The SPS control cards required to do this are:

```
* ASSEMBLE RELOCATABLE
* SYSTEM SYMBOL TABLE
* LIBR
* ID NUMBER 0100
* STORE RELOADABLE
```

The *SYSTEM SYMBOL TABLE card should not be used when assembling PICK because it will produce double definition errors. The number on the ID NUMBER card is the DIM Entry Number for the subroutine being assembled.

The appendix to this paper contains listings and other information which may be useful when converting the users subroutine set.

Electronics

A McGraw-Hill Publication
330 West 42nd Street, New York, N.Y. 10036
Telephone: [212] 971-2645

Lewis H. Young, Editor

June 28, 1965

Dear Mrs. Hall:

Mr. Love of IBM has written us requesting permission for you to reprint "Using a Computer for Circuit Analysis" by Herbert Wall from the November 2, 1964 issue of Electronics.

You have our permission provided the reprint carries the following legend: Reprinted from Electronics, November 2, 1964; copyright 1964 McGraw-Hill, Inc.

Sincerely yours,

Lewis H. Young
Editor

LHY:kf
cc: David J. Love

Mrs. Carol A. Hall
USL Computing Center
University of Southwestern Louisiana
Lafayette, Louisiana

COMPUTER AIDED DESIGN OF INTEGRATED CIRCUITS

by:
Arnold Spitalny
for presentation at
1620 Users Group Meeting
in
Miami Beach, Florida, May 10, 1965

The Norden Division of United Aircraft Corporation has had an active and extensive design automation program for the past 2 1/2 years. We are trying to apply the same thought and effort to development of man-machine systems for performing our engineering work more efficiently that we normally apply to development of radars, display systems, navigation systems, and other complex electronic equipment. Emphasis throughout our program has been placed on development of user-oriented systems in which computer assistance is made conveniently available to engineers in the context of their own problems. These users do not have to become programmers to learn and use the systems we develop.

Since most of our business is in development of electronic equipment, a large part of our design automation effort has been directed toward improved methods for synthesis, analysis, layout, and packaging of electronic circuits. Our early start, rapid progress, and demonstrated capability attracted outside recognition and support in two important areas.

In April 1963, Norden and IBM agreed to combine our separate circuit analysis program development efforts and work together on a comprehensive user-oriented circuit analysis program system. This joint effort resulted in development of the ECAP system which was presented by Mr. Wall. The Norden version is called NORNAP (Norden Network Analysis Program). It includes many features added by Norden after IBM stopped further joint development to prepare ECAP for general release.

Development of an electronic design that performs the proper functions is only one step in the series of activities required for design and production of integrated circuits.

Once a satisfactory electronic design is achieved, the next problem is design of the diffusion and interconnection patterns required to fabricate this circuit within a single silicon chip. This can be a tedious and frustrating job, which increases exponentially in difficulty as we increase the number of components in each circuit. The designer must determine size and shape of all elements, allow for moats and clearance tolerances, arrange the elements into a minimum area rectangle of satisfactory form factor, locate terminals, and route

all interconnections without metalization crossover.

This layout design task is much more difficult than design of a conventional printed circuit board. Errors are not likely to be discovered until after the first devices are made and are rarely possible to correct without repeating the whole manufacturing cycle. The repeated cycle of design, manufacture, test, and modification can be broken by providing a computer-aided design system that helps the device designer to obtain better designs faster, automatically checks the designs for conformance with all process tolerances and design rules, automatically generates a complete set of mask designs corresponding to the checked and approved layout design, and predicts functional performance of the circuit before it is built.

Norden is now developing such a man-machine system under Air Force sponsorship with the assistance of the United Aircraft Research Laboratories. The program is under the direction of Captain H.D. Colwick and Mr. M. Bialer of the Electronics Branch, Manufacturing Technology Division, Air Force Materials Laboratory, Wright Patterson Air Force Base, Ohio.

This system for rapid design of microcircuits is supplemented by the series of Norden-IBM computer programs for analysis of electronic circuits.

These two developments reduce the time required for initial design and manufacture of new microcircuits to less than that needed for conventional circuitry.

Since the basic circuit analysis methods are fairly well known and were discussed by Mr. Wall we will devote most of our time to the newer material on design of circuit layout and interconnection patterns.

Design Cycle Outline and Circuit Analysis

Figure 1 indicates some of the major steps involved in conventional design and development of integral circuits. The full sequence from concept to delivery includes preliminary design, circuit synthesis, circuit analysis, mask layout, mask artwork and cutting, and the many steps of fabrication and test. All the steps shown in ovals were originally performed manually.

Some of the problem areas of the conventional approach are indicated by the feedback loops in Figure 2. The first is to design a circuit that performs the desired function and has parameter values suitable for inclusion within an integral circuit. The conventional experimental breadboard approach requires uses of conventional components and pieces from other integral circuits. Distributed effects and leakage paths within the chip must be estimated and simulated, and other distributed effects caused by the experimental setup must

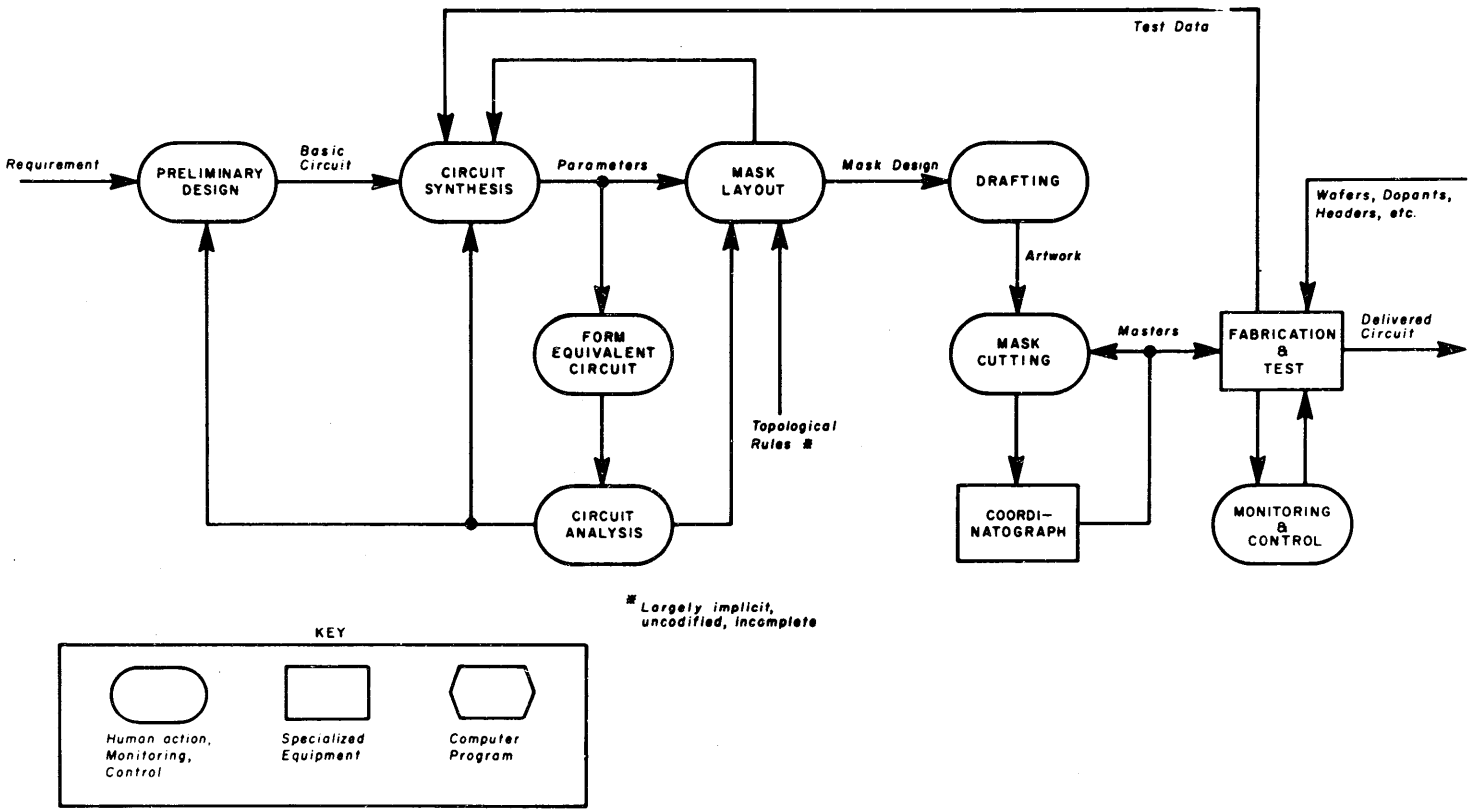


Figure 1. Conventional Design and Development of Integral Circuits

be minimized. Experimental modification of parameters is often much more difficult than in conventional circuits, due to the limited range of integrated circuit elements available for experimental hookup.

This problem is greatly eased by using computer analyses of prospective circuit designs and modifications to supplement the experimental work. A very simple user-oriented language is used to describe circuits to the computer, which can then perform dc, ac, or transient analyses and obtain more information in a few minutes than could be learned in weeks of laboratory tests.

Figure 2 shows a simple equivalent circuit and the corresponding input data that describe the circuit to the computer. Each branch is described by a single IBM card containing branch number, nodes connected to, and values of all circuit parameters. The computer uses these data to set up and solve the matrix formulation of the circuit nodal equations.

Once the circuit is defined to the computer, many analysis routines are available to assist the engineer in analysis and optimization of his circuit. The command instructions for directing the computer to use some of these routines are indicated in Figure 3.

We will now look at the output that results from one of these instructions.

Figure 4 is a photograph of a computer printout. At the top are all the node voltages of the circuit, which are the same as would be measured on a circuit breadboard. Next, we have the sensitivity of every node voltage to variations in every circuit parameter. This information, obtained in less than one second on the IBM 7094 computer or one minute on the 1620 computer, provides more insight into the effect of component variabtion and tolerances than could be obtained in weeks of experimentation. A very simple circuit has been used for this illustration. We routinely handle circuits of up to 20 nodes and 50 branches on the 1620 or 50 nodes and 100 branches on the 7094.

Figure 5 shows how new microcircuits are designed with the system now under development at Norden. The steps shown in hexagons represent computer operations. Functional performance of prospective circuits is determined by computer analysis and optimized before layout begins. Circuit layout is actually performed by automatic computer programs, subject to manual review and modification, and results in automatic preparation of a set of mask artwork.

This system is planned for eventual on-line operation with graphical display and manipulation of circuit layout patterns on a CRT display with a light pen. The present prototype version uses a Calcomp plotter on the 1620 for graphical display and an Orthomat Drafting Machine for automatic preparation of mask artwork. Graphical design manipulations are performed on a 7094 computer in Hartford and

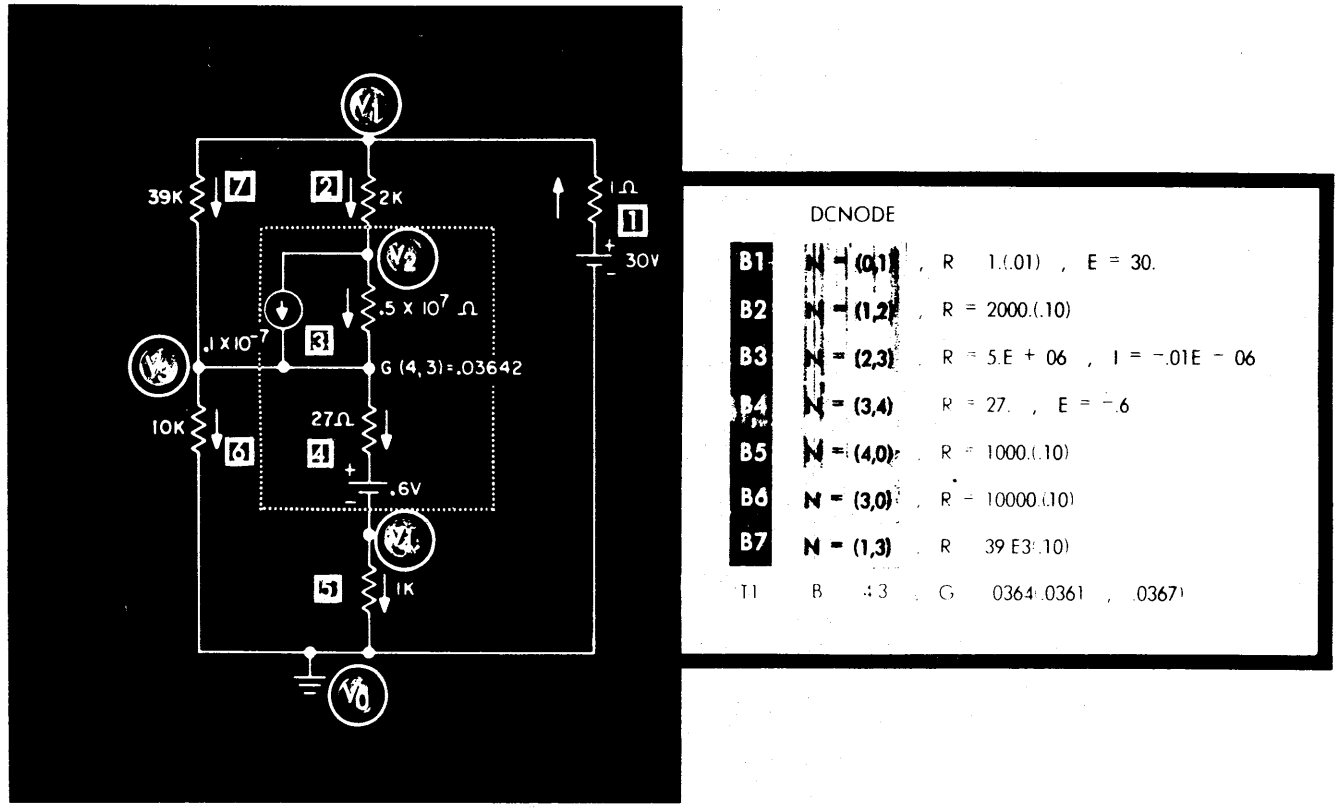


Figure 2. Input Data Preparation

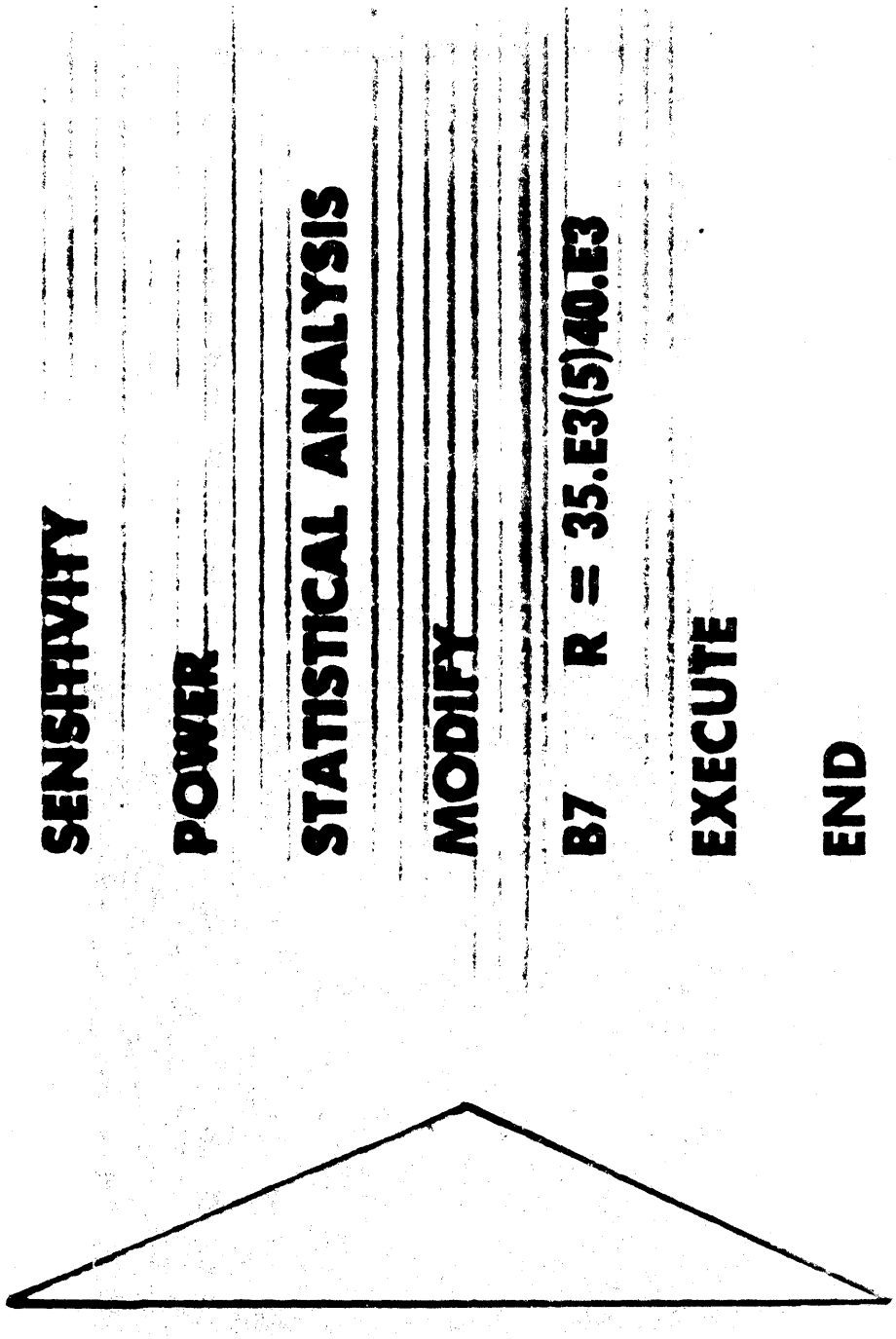


Figure 3. Control Instructions

```
NO. BRANCHES =  7
NO. NODES =  4

 NOMINAL NODE TO DATUM VOLTAGES
    2.999E+01     2.062E+01     5.493E-00     4.764E-00

 SENSITIVITIES
 PERCENT CHANGE IN NODE VOLTAGES FOR A ONE PERCENT CHANGE IN PARAMETERS

R 1
    1.770E-04    -1.683E-04    -1.747E-04    -1.961E-04
R 2
    4.690E-07    -4.530E-01    -2.385E-03    -2.677E-03
R 3
    1.172E-03     3.493E-00    -5.962E-00    -6.693E-00
R 4
    3.623E-06     1.050E-02     2.809E-03    -2.313E-02
R 5
    1.342E-04     3.891E-01     1.040E-01     1.431E-01
R 6
    1.192E-04    -3.564E-01     6.993E-01     7.850E-01
R 7
    1.572E-04     4.076E-01    -7.997E-01    -8.978E-01
G  4, 3
    1.171E-03    -3.491E-00     5.958E-00     6.689E-00
E 1
    9.999E-01     9.509E-01     9.868E-01     1.107E-00
E 4
    1.690E-05     4.900E-02     1.310E-02    -1.079E-01
I 3
    2.502E-09    -7.456E-06     1.272E-05     1.428E-05
```

Figure 4. Computer Printout

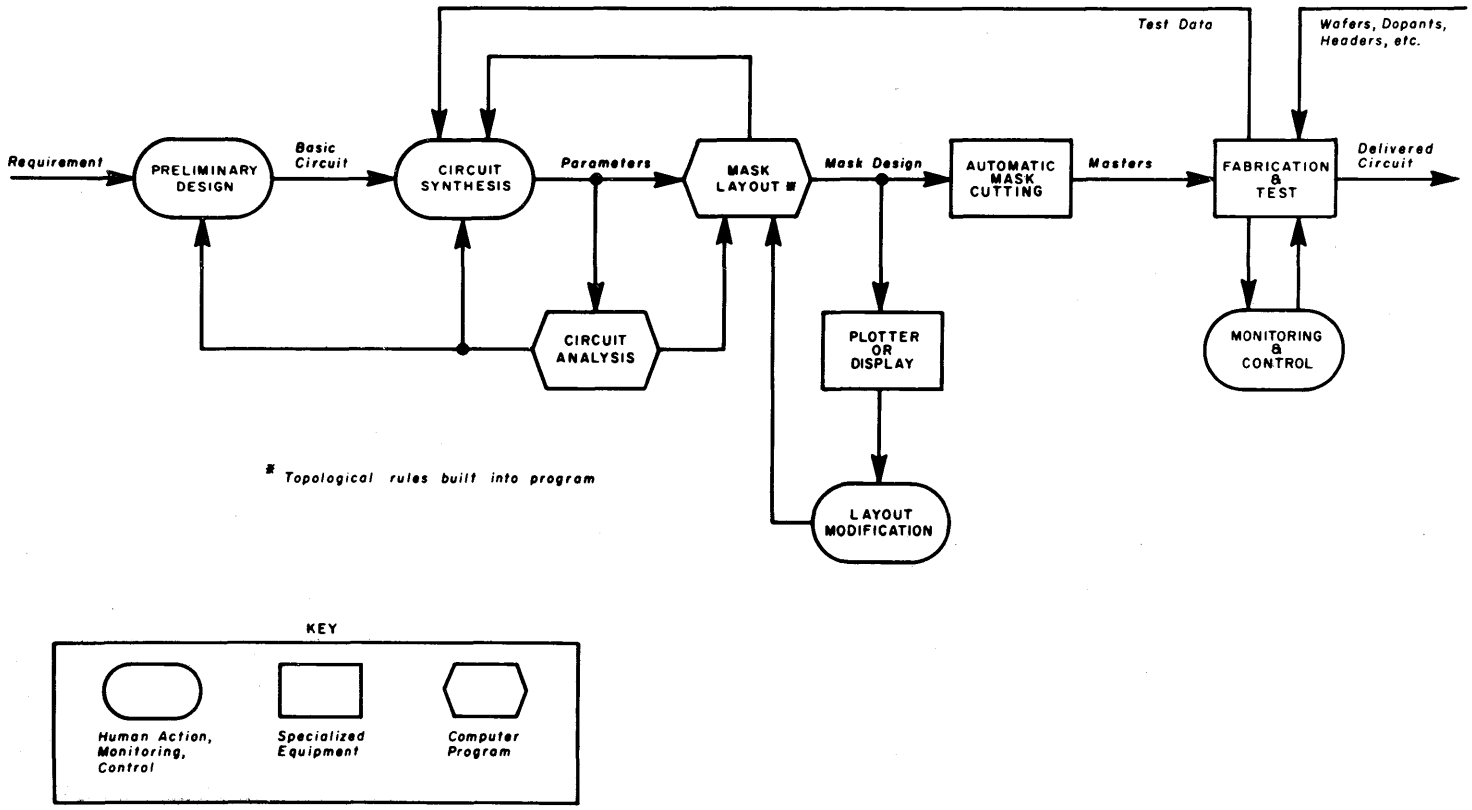


Figure 5. Computer-Aided Design and Development of Integral Circuits.

cards are transceived 70 miles over the phone lines to Norwalk for Calcomp plotting on the 1620 computer.

Circuit Layout Design

Figure 6 is the schematic of a two-stage differential amplifier designed with the prototype computer-aided design system. We will assume that circuit analysis and functional optimization have been completed.

The schematic assigns an initial location for all elements relative to one another, as well as parameter values and interconnections. This information is put into the computer along with the following circuit fabrication requirements:

a) fabrication method

b) sheet resistivity of resistors

c) transistor types

Based on these inputs and a set of design rules developed at Norden, the computer calculates the length and width of each resistor and the area of each capacitor. It also selects from a library of standard shaped transistors the dimensions of the transistor and diode types selected by the designer to meet the electrical requirements of the circuit.

The computer is now ready to plot an initial detailed layout as shown in Figure 7.

This figure and the following figures are photographs of actual computer plotted outputs at various stages in the design process.

This initial layout, Figure 7, is the starting point for a series of man-machine interactions to evolve the final design. The designer looks at this picture and decides to rearrange things a little. He may say, "Let's rotate this transistor and move it up here, bring the other one as close as possible below it, put R10 on top to join it with R15, move R_2 up and put R_{12} alongside it, do the same to the other side to maintain symmetry, and take a look at what that gives us." (Figure 8)

These instructions are coded in a simple user-oriented language, punched on cards, and fed to the computer.

The complete vocabulary of 23 command instructions provides a capability for assigning any desired orientation to any circuit element and locating it in any desired position relative to the other elements. Provisions are included for routing interconnections, bending and joining resistors, shaping capacitors, adding and deleting moats, and performing other routine design functions.

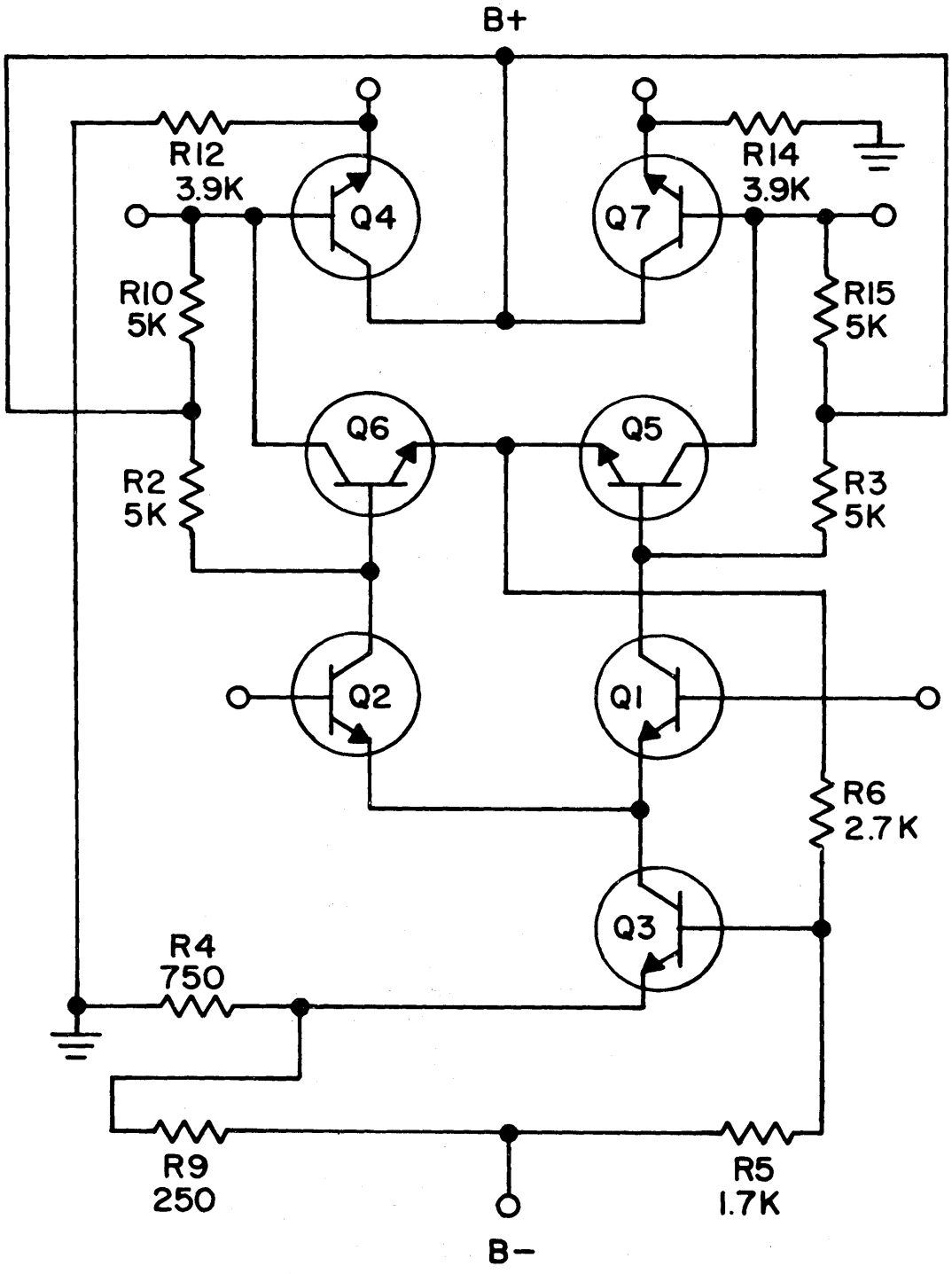


Figure 6. Two-Stage Differential Amplifier

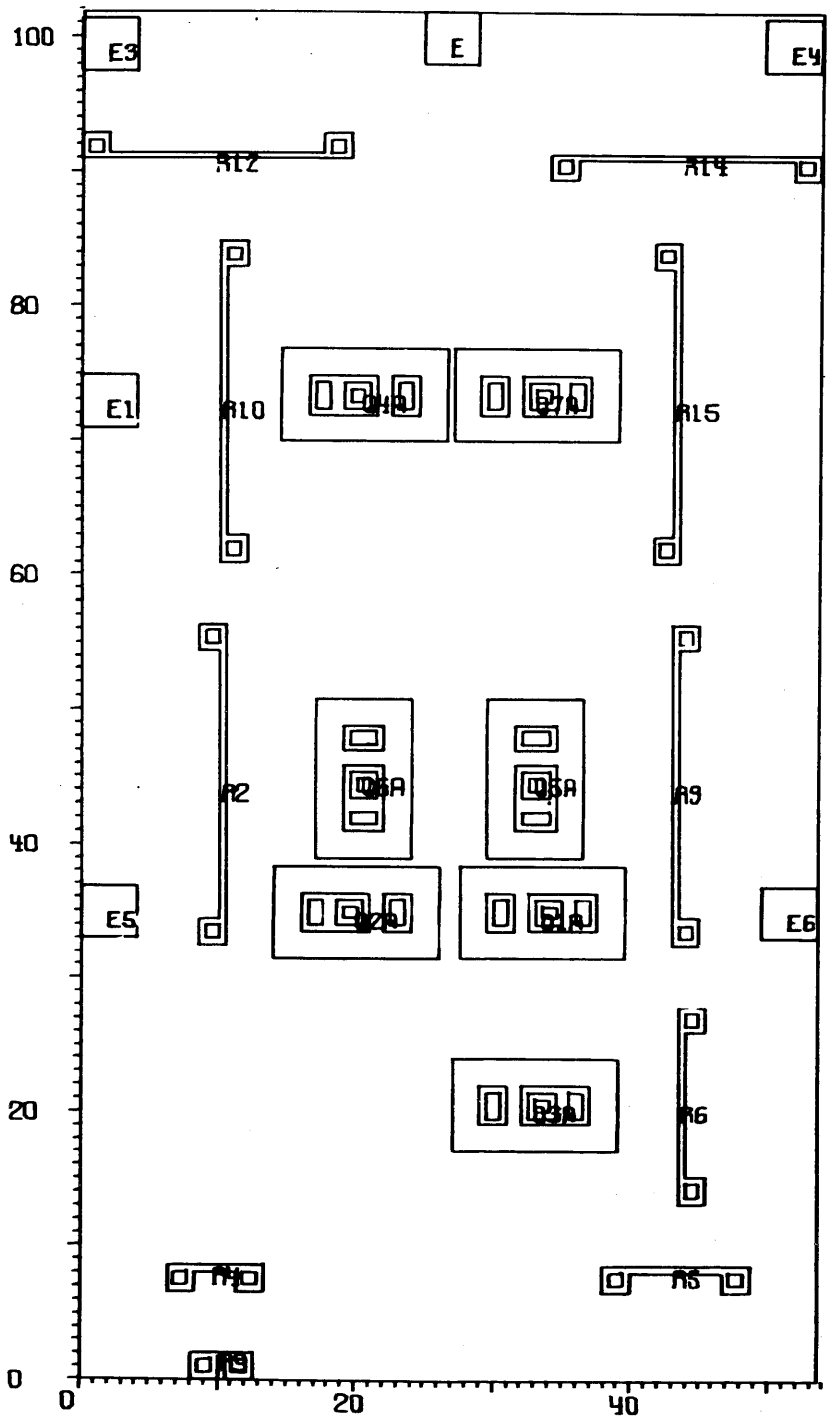


Figure 7. Initial Layout

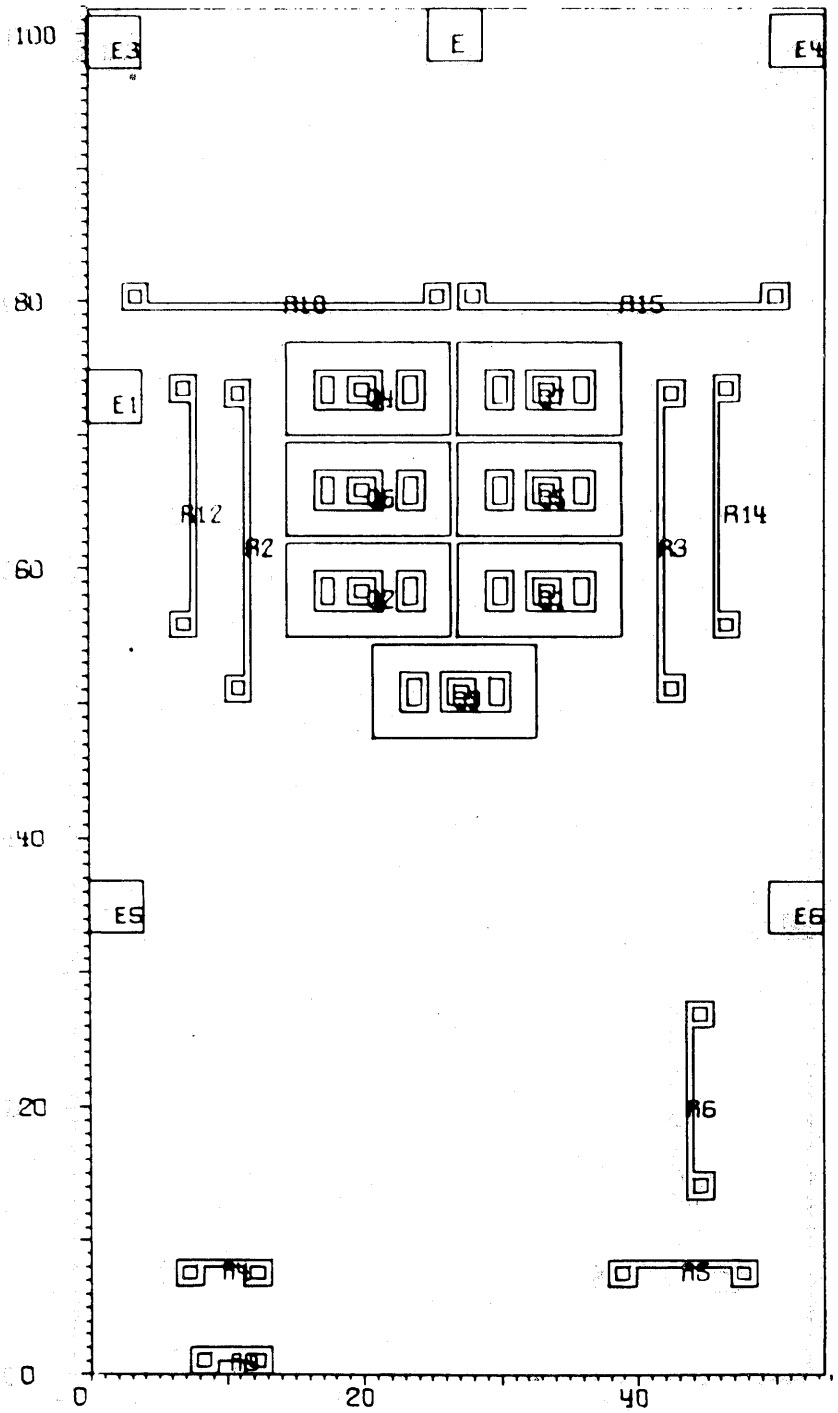


Figure 8. First Modification

Figure 9 and 10 show successive layout improvements illustrating some of these capabilities.

A translator program interprets the instructions and calls in manipulative programs to change the layout in accordance with the deisgner's instructions and the appropriate programmed or recorded rules and tolerances. In the on-line system, the design modification language will be converted from cards to light pen and pushbutton operations.

Each succeeding transformation is graphically displayed by the computer to aid the designer in planning his next design step. The process of looking at a trial layout and instructing the computer to make changes is repeated until a satisfactory layout is achieved (Figure 10). On the circuit layout is accepted, the designer directs the computer to design diffusion masks from the final composite two-dimensional design. The computer then punches control instructions on paper tape to make each mask on an accurate, high resolution drafting machine. Figures 11, 12, 13, 14 are photographs of computer-plotted mask designs for the differential amplifier.

Interconnection Routing

Routing of interconnection metalization is an important consideration in planning location of diffused elements and is a difficult job for large circuits, even after a favorable layout is obtained. It is even more difficult to take a circuit diffusion pattern designed for one application and reconnect it to form a completely different circuit.

Figure 15 is a computer-designed interconnection pattern for an error amplifier. More than 50 different circuits have been designed with this same diffusion pattern by rearranging the interconnections.

A maze-solving computer program is used to route interconnections, automatically avoiding all contacts of other nodes and all previously routed interconnections. The resulting pattern can vary, depending on the routing sequence. When a connection is completely blocked by previously routed connections, the computer automatically changes the sequence and tries again. The pattern shown here was successfully routed on the third automatically sequenced trial.

Computer System Design

Every effort is being made to modularize the computer system design so that changes in integrated circuit technology and conversion to an on-line system will result in minimum additional programming effort.

As the technology of integrated circuit design changes, new programs can be appended to the system and present programs modified

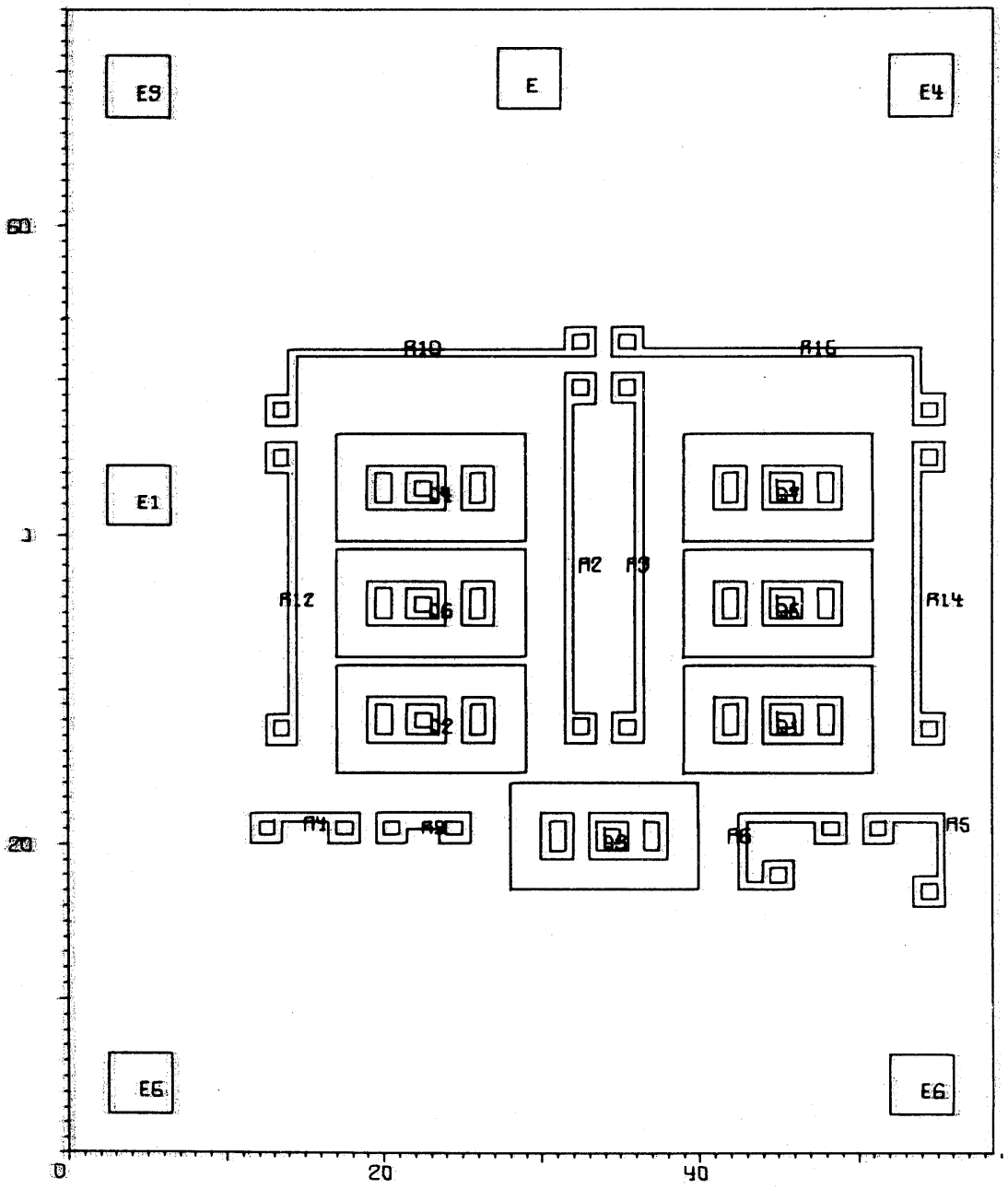


Figure 9. Final Arrangement

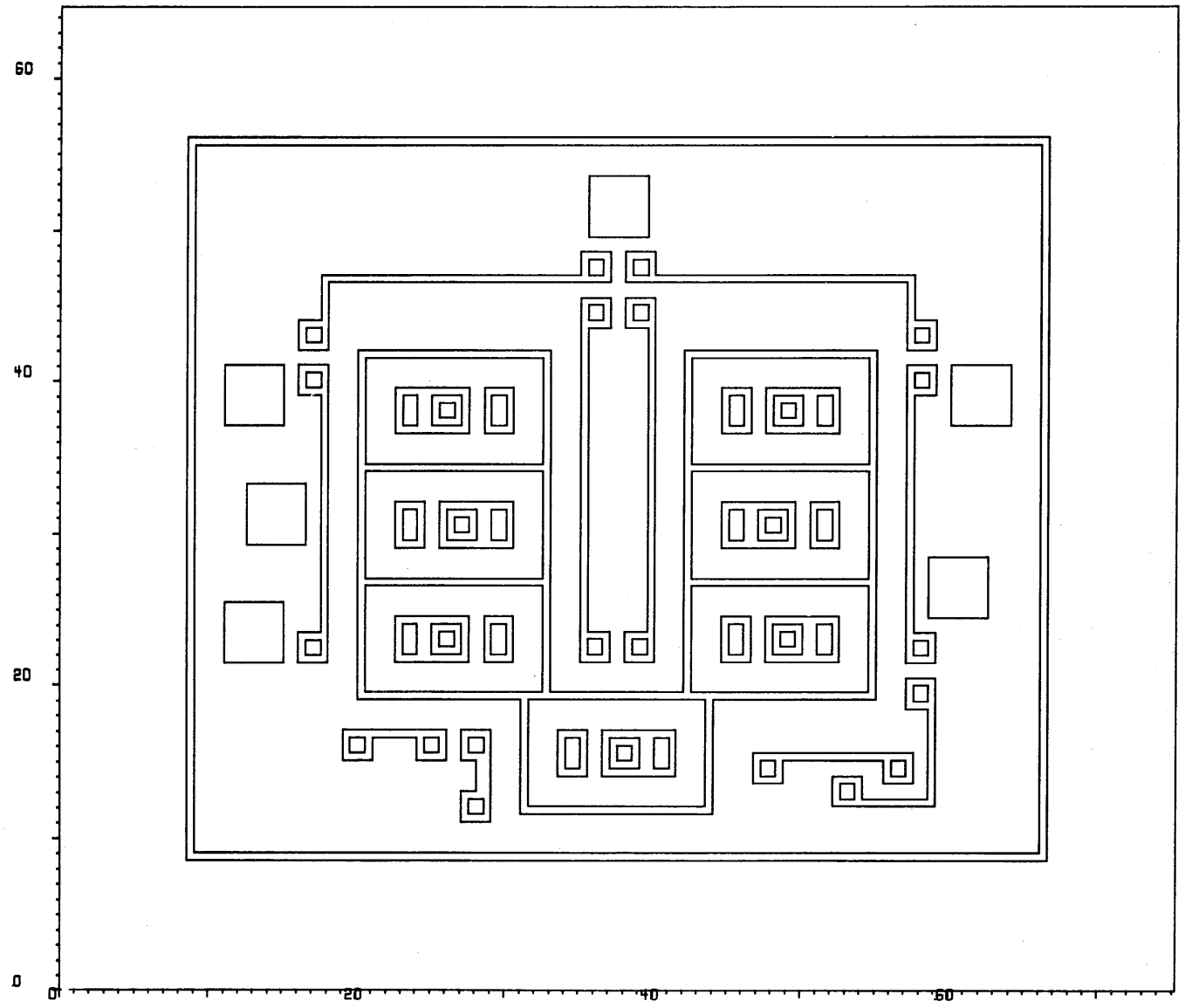


Figure 10. Final Detailed Layout - 2-Stage Differential Amplifier

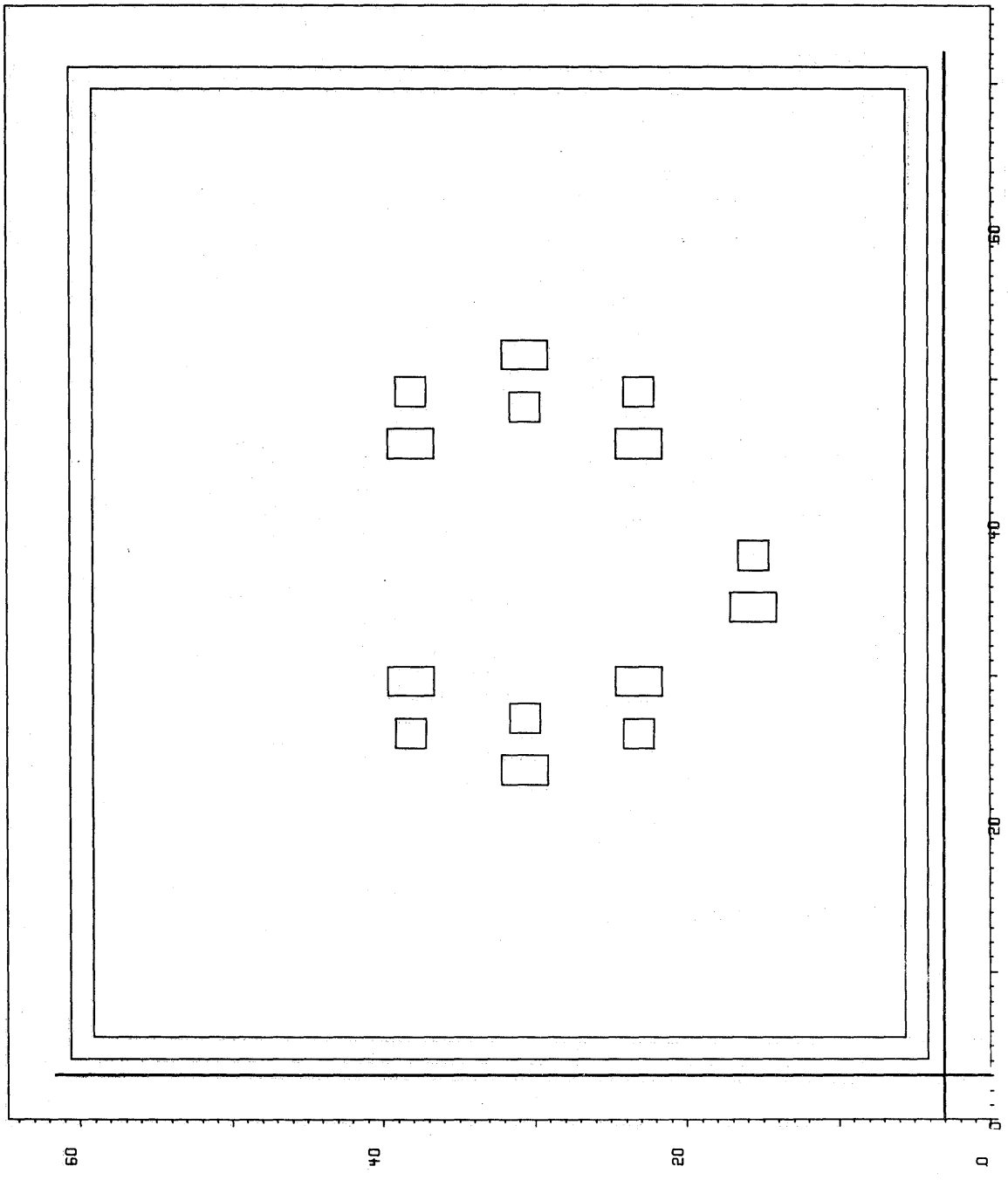


Figure 11. Emitter Mask

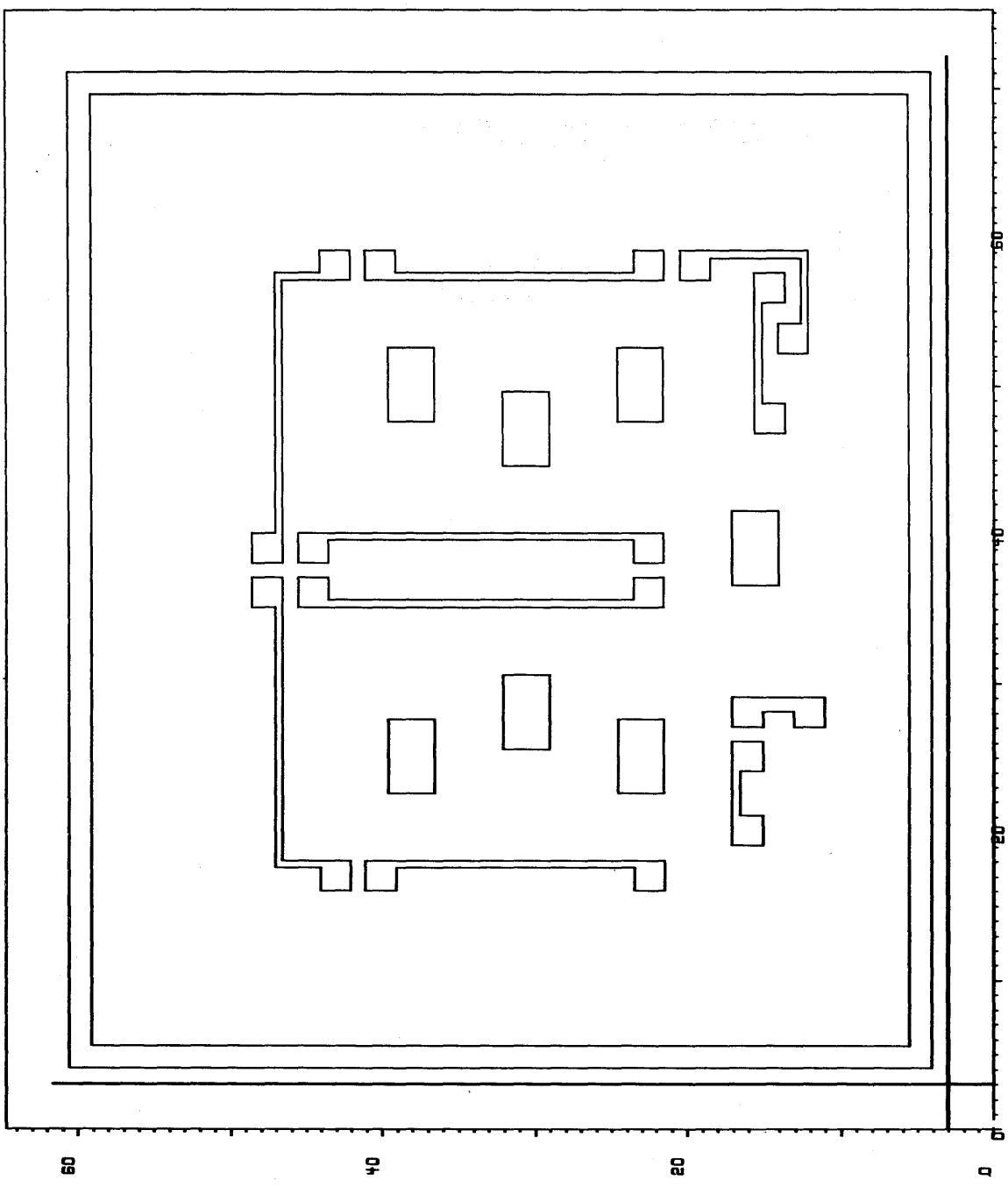


Figure 12. Base Mask

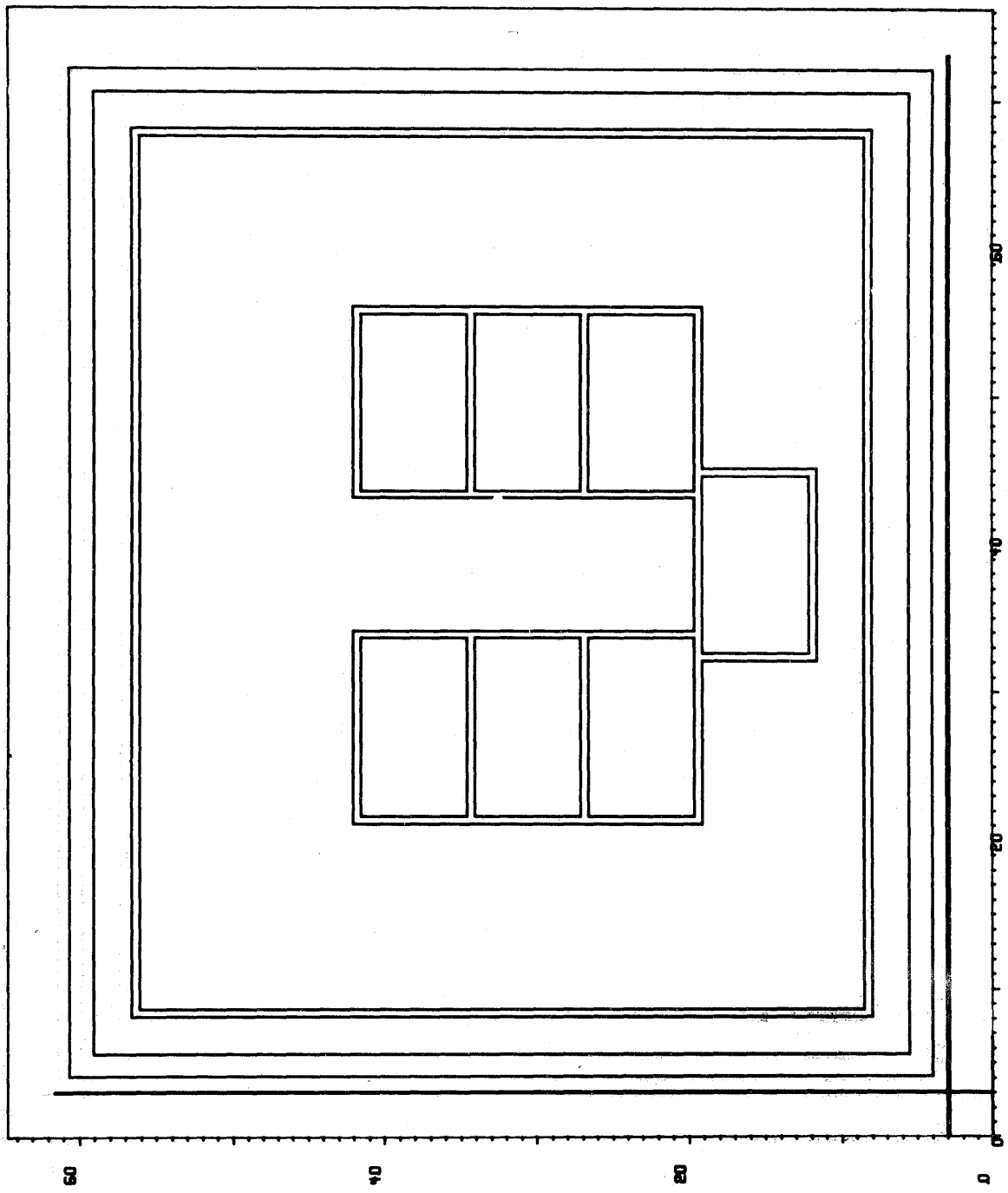


Figure 13. Moat Mask

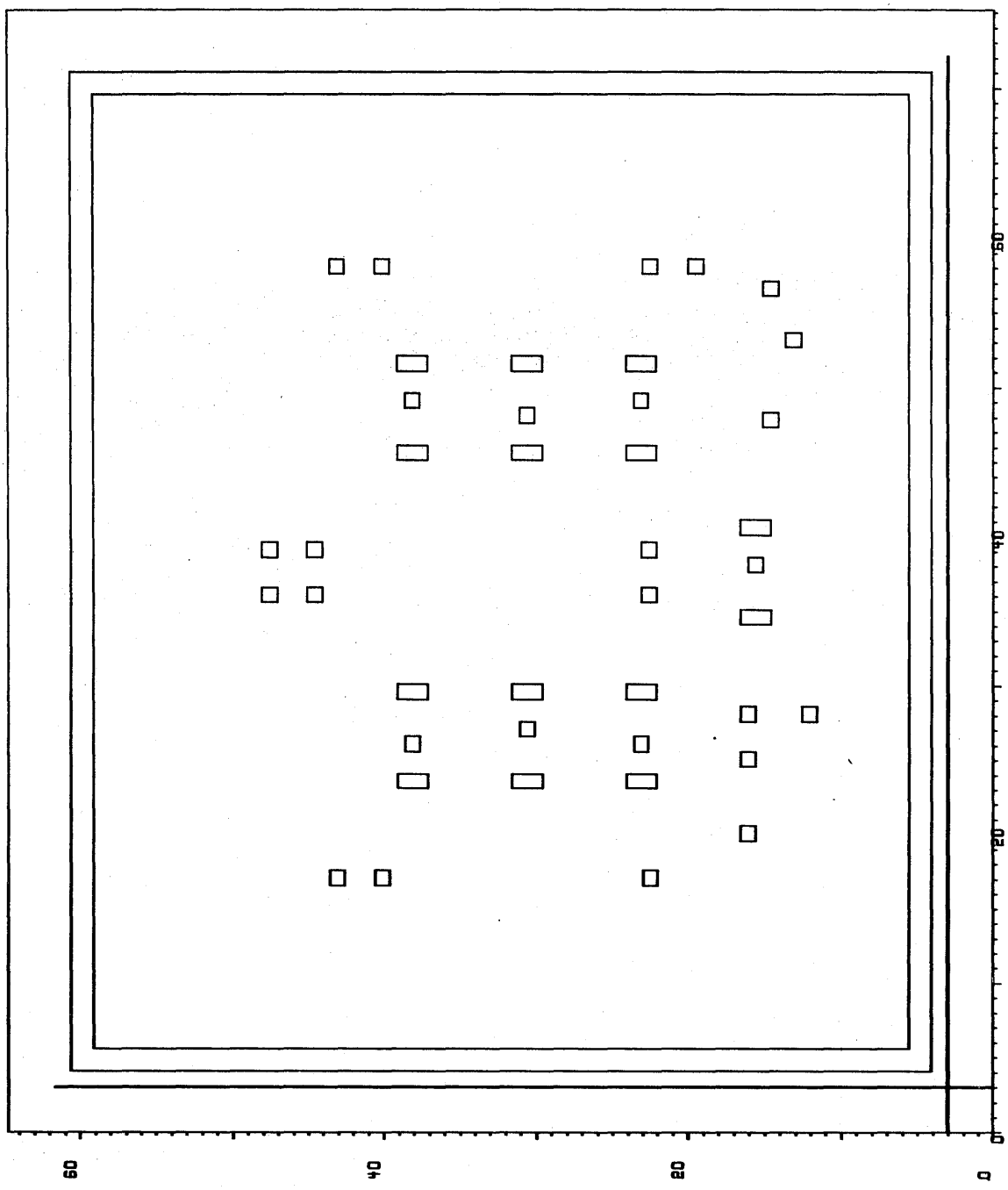


Figure 14. Contact Mask

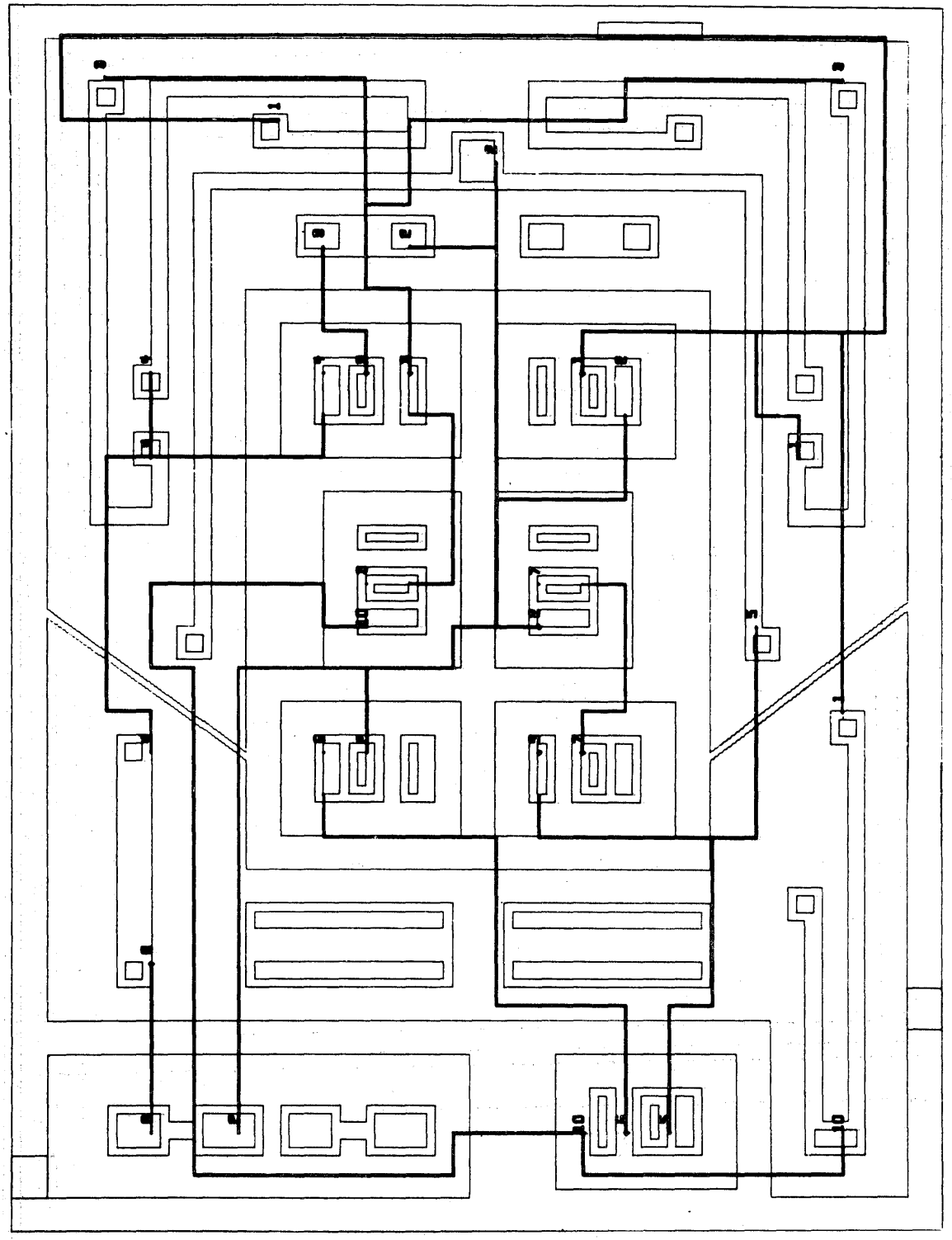


Figure 15. Final Routing

without the expense of disturbing the logic of the operating system or user program links not requiring modification. Conversion to an on-line graphic system will be realized by simply expanding the processor function of I/O control.

All programs are written in FORTRAN to the maximum extent feasible in order to make this system machine-independent. It will be able to run on any computer system that has the necessary capacity, graphical display terminals, and basic software. Changes in computer technology can thus be readily accommodated, as well as changes in integrated circuit technology.

Status and Significance

The prototype computer-aided design system is now in final development and initial experimental use. The design rules programmed into this system have been thoroughly tested by manual application on all integrated circuits designed by Norden in 1964. Manual operations are gradually being replaced by computer operations as the programs become available. Design layout time for a complex circuit has already been reduced from several weeks to a few days. With the planned on-line computer-aided system, it will be reduced to hours or minutes. We believe that this system and especially the on-line version to follow it represent a significant breakthrough in the design bottleneck.

Since this system is being developed under Air Force funding, it is not considered proprietary to Norden. Computer programs and instructions will be made available to the industry by the Air Force when system development is completed. If you want additional information, or are interested in the possibility of using this system in your own facility, you should contact our Air Force Project officer;

Mr. Max Bialer
Electronics Branch
Manufacturing Technology Division
Air Force Materials Laboratory
Wright Patterson Air Force Base, Ohio

D. C. AENEAS

(DIRECT CURRENT AUTOMATIC ELECTRIC NETWORK ANALYSIS SYSTEM)

by

Heberto Pachon
Mathematical Analysis Section
Research Laboratory
Automatic Electric Laboratories, Inc.
Northlake, Illinois

Presented at the
"1620 Users Group Meeting"
in Miami Beach on
May 10, 1965.

D. C. AENEAS

1. Introduction

This is a linear graph theory application paper describing in general terms a computer program developed for use on the IBM 1620 Data Processing System, and capable of performing automatic D. C. analysis of electronic circuits.

The program carries the name of D. C. AENEAS which stands for Direct Current Automatic Electric NEtwork Analysis System, and it incorporates the following features:

1a It reads and decodes the geometrical (topological) configuration, the component characteristics, and the component values of a D. C. network from easily coded statements punched on tabulating cards.

1b It generates the network equations and solves for nominal branch currents, nominal node voltages, and nominal power dissipations.

1c It computes branch currents sensitivity factors, and node voltages sensitivity factors.

1d It computes node voltage standard deviations.

1e It computes minimum and maximum worst case analysis for branch currents and node voltages.

1f It allows the user to modify the values of one or two components in the network in order to generate tables of branch currents or node voltages as functions of the components being modified. These changes can be effected without a complete recalculation of the problem from start to finish.

Features 1b - 1f are optional, and under user's control.

To use this program, the engineer draws an equivalent circuit representing the circuit he wishes to analyze. He then numbers the nodes and the elements of the circuit, and at the same time he assigns arbitrary current and voltage references to the circuit elements. A special input language, whose statements are in one-to-one correspondence with the circuit diagram so far developed, is then used to describe to the program the topological configuration, the component characteristics, and the component values of the circuit. The engineer also uses this input language in order to specify what is to be calculated and what is to be punched as output.

All the information describing the network is punched on tabulating cards, and these cards are used as an input to the D. C. AENEAS program. The program will go through all the calculations necessary, such as generating the circuit equations and solving these equations, and will prepare a report on punched cards. This report consists of a duplication of the input cards followed by all numerical answers requested by the engineer. These cards must be listed in order to obtain understandable answers.

The program can operate on networks containing linear resistors, constant voltage generators, constant current sources, and linearly dependent current sources* only. Therefore, when the network for analysis contains active devices (such as transistors, diodes and vacuum tubes) equivalent circuits must be obtained before the engineer can make use of the D. C. AENEAS program. Fortunately, adequate equivalent circuits exist for these types of devices; hence, most electronic networks can be analyzed, D. C. wise, by means of this program.

2. Subprograms

D. C. AENEAS consists of six programs which operate upon a common area within the computer memory called the Network Memory Map, and which are capable of performing features 1a - 1f mentioned under section 1. These programs are as follows:

2a Input Program

All the necessary information pertaining to the network under analysis is supplied by the user to the D. C. AENEAS program by means of statements comprised in what is called the input language. These statements are punched on tabulating cards which are read and decoded by the input program. In the process of decoding, the input program also fills in the Network Memory Map and checks for possible coding errors.

2b Topological Program

The topological program makes use of part of the information contained in the Network Memory Map; more exactly, it makes use of the information describing the connectivity (topology) of the network in order to generate a network tree, a network co-tree and finally a network circuit matrix. The program

---

* By linearly dependent current source we mean a current source whose value depends on the value of the current through one resistor in the circuit; the dependence is linear.

also checks for errors and gives the user the option to type the different connectivity matrices associated with the tree and co-tree, as well as the final circuit matrix.

2c Calculator Program

The circuit matrix, together with the network component values contained in the Network Memory Map, are used by the calculator program in order to generate the loop equations for the network, and thereafter to solve for all the network nominal constraints requested by the user.

2d Sensitivity Program

The sensitivity program computes the partial derivatives of all unknown variables (branch currents and node voltages) with respect to all of the network parameters, and stores information for further use in connection with the calculation of worst cases or standard deviations.

The program computes partial derivatives by means of closed formulas, a method which seems to be faster than the calculation by approximation method.

2e Common Program

The common program is basically a duplicate of the calculator program; that is, its main function is to set-up loop equations and solve for branch currents or node voltages. However, this program contains two subroutines that correspond to the processing of the worst case and modify control statements.

Using the signs of partial derivatives obtained by the sensitivity program, the worst case subroutine rearranges the values of the parameters within the Network Memory Map and branches to the generation and solution of equations subroutine to determine the tolerance extremes of branch currents and/or node voltages.

Using the data corresponding to the modify control statement, the modify subroutine modifies the value, or the values of the corresponding parameters, and branches to the generation and solution of equations subroutine to evaluate branch currents or node voltages.

2f Standard Deviations Program

Using the values of the partial derivatives, the standard

deviation program computes the standard deviations of all unknown node voltages.

3. General Comments

The first three programs mentioned in section 2 accomplish the true electrical analysis of the circuit problem.

The input language developed to communicate with the program is oriented towards the electrical engineer's use and requires only a few minutes to learn. Additional statements may be added to form an input language useful, not only to D. C. analysis programs, but to A. C. analysis and transient analysis programs as well.

The method of obtaining the circuit matrix was obtained from the work done by Professors M. B. Reed and S. Seshu (1), but we extended their work in the generation of the loop system of equations when current dependent sources are present.

The simulation of the electrical network by means of the equivalent circuit on the digital computer is a very important achievement (2) because, if such a simulation is possible, a great saving is effected in cost and time when certain solutions to vast problems are desired. In addition, such simulation transforms the digital computer (IBM 1620 in this case) into a large analog computer having great flexibility and high degree of accuracy.

The last three programs mentioned in section 2 give the engineer a way to predict possible troubles due to changes in the components within the network under analysis. When properly used, these programs can also lead the engineer to the most reliable selection of network component tolerances.

The closed formulas developed for the evaluation of partial derivatives seem to be a considerable improvement over existing approximation methods. We have made no attempt to determine exactly how good our method is, but we have estimated that it will improve the speed of calculation by about a factor of 8.

Finally, we would like to point out that since the program is a rather complicated combination of matrix operations, the use of the PUC-R2 compiler (3) was extremely helpful and time-saving in obtaining the final symbolic language source deck for the program.

4. Example

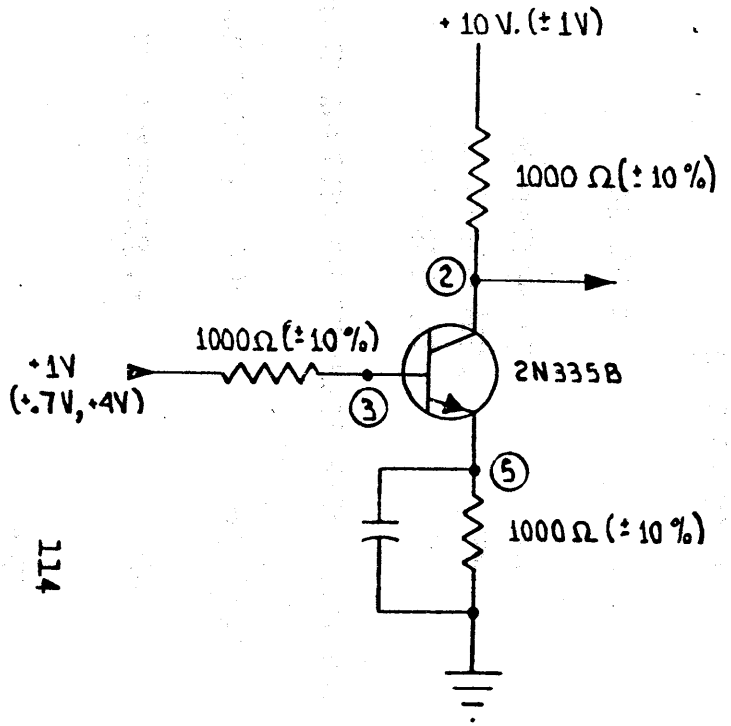
The attached example, found in the SPARC (4) report illustrates the use of the language developed for communication with the D. C. AENEAS program, and it also demonstrates part of the calculations that the program can perform.

Heberto Pachon 4/21/65

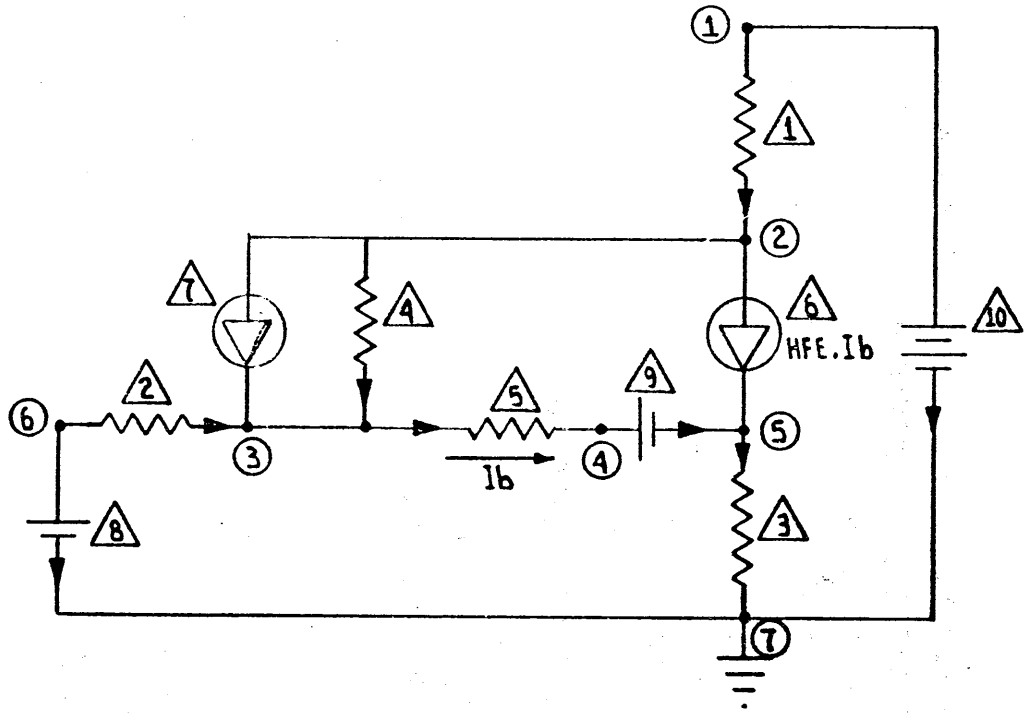
Heberto Pachon

SAMPLE PROBLEM

CIRCUIT SCHEMATIC



EQUIVALENT CIRCUIT



○ = NODE NUMBER

△ = ELEMENT NUMBER

→ = ELEMENT ORIENTATION

R4 = 3.03 MΩ (1.82 MΩ, 9.09 MΩ)

R5 = 1685Ω (670Ω, 2700Ω)

HFE = 56 (19, 93)

V9 = +.388V. (.28V., .6V.)

CALCULATE:

1. NOMINAL NODE VOLTAGES AT NODES 2, 3, & 5
2. PARTIAL DERIVATIVES OF NODE VOLTAGES (2, 3, 5) WITH RESPECT TO ALL OF THE NETWORK ELEMENTS.
3. ALL NODE VOLTAGES AS THE hfe PARAMETER OF THE TRANSISTOR CHANGES FROM 55.44 TO 56.56 IN STEPS OF .56. THE VALUES SO OBTAINED CAN BE USED TO CHECK THE VALUES OF THE PARTIAL DERIVATIVES WITH RESPECT TO ELEMENT NO. 6 OBTAINED IN 2 ABOVE.

4/21/65

```
C       D.C. AENEAS TEST - PROBLEM NO. 1 - SPARC PROGRAM EXAMPLE
C       AUTOMATIC ELECTRIC LABORATORIES - RESEARCH
C       DANIELLE DONNELLY - HEBERTO PACHON
C       APRIL 19, 1965
C
E1      R( 1, 2) = 900., 1K, 1.1K
E2      R( 6, 3) = .9K, 1000 , 1100.
E3      R( 5, 7) = .9E+03, 1K
E4      R( 2, 3) = .182E+07, .303E+07, .909E+07
E5      R( 3, 4) = 670, 1685, 2700
E6      J( 2, 5) , HFE( 5 ) = 19, 56, 93
E7      J( 2, 3) = .1E-06, 2.E-06, 4.E-06
E8      V( 6, 7) = .7, 1., 4.
E9      V( 4, 5) = .28, .388, .6
E10     V( 1, 7) = 9, 10, 11
        RV = 7
        NODE VOLTAGES ( 2,3,5 )
        SENSITIVITY
        MODIFY E6 = 55.44, .56, 56.56
        NODE VOLTAGES
        FIN
```

NODE VOLTAGES

NODE NO.	VOLTAGE
02	.94165220E 01
03	.99444570E 00
05	.58903298E 00

PARTIALS OF NODE VOLTAGES WITH RESPECT TO VOLTAGE ELEMENTS

NODE NO.	PTL. WRT. ELEMENT NO. 08
02	-.93703124E 00

03	.98262250E 00
05	.95440879E 00

NODE NO.	PTL. WRT. ELEMENT NO. 09
02	.93767029E 00
03	.17053371E-01
05	-.95472366E 00

NODE NO.	PTL. WRT. ELEMENT NO. 10
02	.99936100E 00
03	.32419085E-03
05	.31488247E-03

PARTIALS OF NODE VOLTAGES WITH RESPECT TO RESISTIVE ELEMENTS

NODE NO.	PTL. WRT. ELEMENT NO. 04
02	.17763444E-08
03	-.90110884E-09
05	-.87523562E-09

NODE NO.	PTL. WRT. ELEMENT NO. 05
02	.96898023E-05
03	.17622804E-06
05	-.98660302E-05

NODE NO.	PTL. WRT. ELEMENT NO. 01
02	-.58310575E-03
03	-.18915843E-06
05	-.18372718E-06

NODE NO.	PTL. WRT. ELEMENT NO. 02
02	.52045994E-05
03	-.54578290E-05
05	-.53011204E-05

NODE NO.	PTL. WRT. ELEMENT NO. 03
02	.55231873E-03
03	.10044998E-04
05	.26669260E-04

PARTIALS OF NODE VOLTAGES WITH RESPECT TO TR. CURR. ELEMENTS

NODE NO.	PTL. WRT. ELEMENT NO. 06
02	-.63750700E-03
03	.17287788E-03
05	.46462900E-03

PARTIALS OF NODE VOLTAGES WITH RESPECT TO CN. CURR. ELEMENTS

NODE NO.	PTL. WRT. ELEMENT NO. 07
02	-.19363921E 04
03	.98229830E 03
05	.95409393E 03

PARAMETER VARIATION OUTPUT

ELEMENT NO. 06 = .55440000E 02

NODE VOLTAGES

NODE NO.	VOLTAGE

01	.10000000E 02
02	.94168820E 01
03	.99434800E 00
04	.97677034E 00
05	.58877034E 00
06	.10000000E 01

PARAMETER VARIATION OUTPUT

ELEMENT NO. 06 = .56000000E 02

NODE VOLTAGES

NODE NO.	VOLTAGE
01	.10000000E 02
02	.94165220E 01
03	.99444570E 00
04	.97703298E 00
05	.58903298E 00
06	.10000000E 01

PARAMETER VARIATION OUTPUT

ELEMENT NO. 06 = .56560000E 02

NODE VOLTAGES

NODE NO.	VOLTAGE
01	.10000000E 02
02	.94161680E 01
03	.99454160E 00
04	.97729076E 00
05	.58929076E 00
06	.10000000E 01

REFERENCES

1. Seshu, S. and Reed, M. B., "Linear Graphs and Electrical Networks," Addison-Wesley Publishing Company, Inc., Reeding, Massachusetts (1961), pages 117-144.

2. Bashkow, T. R. and Deser, C. A., "Digital Computers and Network Theory," Wescon Convention Record, Vol. 1, part 2, (1953), pages 133-6.

3. Domingo, C. and Rodriguez, F. G., "Central University of Venezuela Processor" (PUC-R2), translated from Spanish by H. Pachon, Automatic Electric Laboratories Report.

4. Kleiner, C. T. and Johnson, E. D., "Sparc-System of Programs for Development and Analysis of Components and Subsystems by Recomp - II," Autonetics Report, pages 1-47.

TRAVERSE ADJUSTMENT BY WEIGHTED LEAST SQUARES

by

Robert L. Kenngott, Supervisor, Computer Center
Chas. H. Sells, Inc., Civil Engineers + Surveyors
409 Manville Rd., Pleasantville, N.Y., 10570

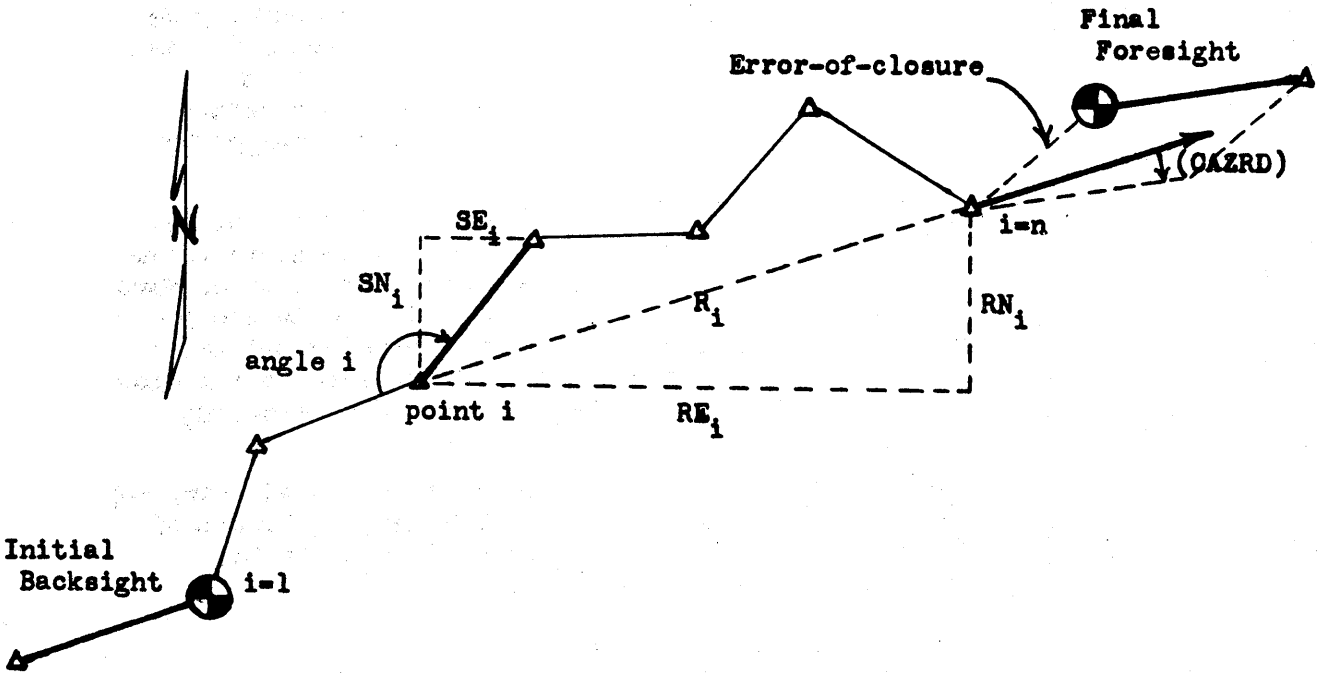
In the author's belief, the entitled FORTRAN program represents the first practical application of the unrestricted power of least-squares to the adjustment of all elements of a survey traverse. To ease the burden of manual computation, conventional methods of traverse adjustment have either been purely artificial (compass and transit rules) or by limited approximation of a least-squares solution (Crandall[1] and Belote[2]).

Above all, any method of adjustment should satisfy at least one criterion; it should never introduce corrections which greatly exceed the standard deviations in the corresponding observations of angles and distances. Conventional methods fail this test and, not infrequently, create artificial map distortions.

Traverse adjustment by weighted least-squares distributes corrections which are in proportion to their effectiveness in reducing the error-of-closure as well as in correct proportion to the individual standard deviations in measurement.

The program adjusts both angles and distances taken directly from field notes for a simple traverse between two known points with terminal azimuths. Angular input may be as deflections, included angles, or azimuths. Terminal points may be co-incident. It is written as a two-pass program for a minimum 1620 card system and will handle an unlimited number of traverse lines. However, all quantities having the dimensions of length are restricted to eight significant figures with three decimal places.

The general method of solving for corrections is indicated in highly abbreviated form by the diagram and equations next below.



Let-

CAZRD = correction to close azimuth (radians)

$(CA1RD)_i$ = correction to angle i for azimuth closure (radians)

$(CA2RD)_i$ = correction to angle i for position closure (radians)

$(CATRD)_i$ = total correction to angle i (radians)

$(SDARD)_i$ = standard deviation in measurement of angle i (radians)

$(CSFT)_i$ = correction to distance S_i for position closure (feet)

$(SDSFT)_i$ = standard deviation in measurement of distance S_i (feet)

Then, corrections are computed as follows:

$$(CA1RD)_i = (CAZRD)[(SDARD)_i^2/\Sigma_1^n(SDARD)_i^2]$$

$$(CA2RD)_i = (SDARD)_i^2[\ U - V(RE_i) + W(RN_i)\ ]$$

$$(CATRD)_i = (CA1RD)_i + (CA2RD)_i$$

$$(CSFT)_i = [(SDSFT)_i^2/S_i][\ V(SN_i) + W(SE_i)]$$

-where U, V, and W are constant co-efficients (Gaussian correlatives) computed by conventional least-squares procedure subject to the conditions for azimuth and position closure.

The quantities subject to least-squares minimization are the ratios of adjustments per the corresponding standard deviations,

$$\sum_1^n \left[\left(\frac{CATRD}{SDARD}\right)_i^2 + \left(\frac{CSFT}{SDSFT}\right)_i^2\right]$$

-from which it can be seen that weighting of the adjustments is controlled through input of the standard deviations (e.g.,zero standard deviation results in a zero adjustment). It may be done individually by direct input or by coded reference to stored subroutines according to the method of observation (repeating transit, direction theodolite with interchangeable target kit, steel tape, Geodimeter, etc.). Subroutine parameters are read in at the start of the program and may be changed by substituting new parameter cards.

Output begins with tabulation of errors-of-closure (azimuth closure, raw position closure, and position closure after angle adjustment to close in azimuth). At the same time, confidence limits in the errors-of-closure are predicted from the root-sum-of-squares of the partial effects due to individual standard deviations. Should an error-of-closure exceed the 95% limit (odds against it are 20/1), the unbalanced traverse is computed and tabulated without further adjustment. This is to aid in locating gross mistakes.

Normally, the output proceeds with tabulation of the adjusted traverse, including a listing of individual corrections and their weights (standard deviations) as well as adjusted azimuths, distances, and co-ordinates.

Finally, because surveyors are most likely to evaluate traverse adjustment by weighted least-squares from a comparison of results; sense switch control will permit alternative adjustment by compass rule, transit rule, or Crandall's method (COGO least-squares).

This program will be submitted to and should be available from the USERS library by the third quarter of 1965. Until then, inquiries addressed to the author will be answered as promptly as possible.

References:

1. C. L. Crandall, "The adjustment of a transit survey-" Proceedings, ASCE, v.26, p.1164, Dec.,1900

2. E. S. Belote, "Adjustment of traverse", Transactions, American Geophysical Union, v.27, no.III, p.307, June, 1946

A 3-DIMENSIONAL COGO SYSTEM
For
HIGHWAY GEOMETRICS

By
ELECTRONIC DATA PROCESSING, INC.
James W. Madden, P.E.
Elliott C. Friedwald

Our 3-D Cogo system came about from a heavy influx of interstate highway design work by Florida consulting engineering firms.

It was developed initially to compute bridge deck elevations for bridges on circular and vertical curves in various stages of superelevation. We have since found it a useful tool in computing grade lines for pavements, medians and side ditches.

We were assisted by the consulting engineering firm of Beiswenger-Hoch-Arnold and Associates of Akron, Jacksonville and Fort Lauderdale in the preparation of the program logic contained in the elevations routine. We believe we have a very workable and practical Cogo system as it is currently in use by nine engineering firms in this area.

In essence, 3D Cogo is simply an SPS plug deck addition to the standard IBM Cogo package.

The system as developed by us is for a 20K card 1620 with 1443 printer. It is also available on card output for possible 407 listing.

My portion of this presentation will be limited to an explanation of the various commands available in the elevations plug deck along with some typical running times.

Mr. Elliott Friedwald, our Chief Programmer, will discuss some of the problems he encountered in revising Cogo for card or printer output and will also discuss the operation of the Cogo system in relation to programming problems.

The operating procedure for the elevations routines is identical to the other Cogo plug decks. The attached five pages are excerpts from our Cogo manual and explain the elevations plug deck and its related commands.

The procedure generally used is to compute the X and Y coordinates of each point for which a station-elevation-offset is required. Next, the elevations plug deck is called into memory and the horizontal curve, vertical curve and superelevation criterea are specified. After this initial set up, the coordinate or elevation of any point contained in the X,Y coordinate table may be computed.

If you will turn your attention to the Cogo elevations manual, we will discuss each command in the order presented.

PROGRAMMING COGO ELEVATIONS

The present version of Cogo elevations is the result of a great deal of experimentation in programming.

When the problem was first presented, the method of attack chosen was the straightforward way--the writing of a Cogo plug deck according to the specifications in the Cogo manual. This was attempted and much to our chagrin, we discovered that we had memory capacity problems in a big way. We had an overflow of several thousand digits.

We were then faced with the decision to revise the program specifications radically, making the program less useful or harder to handle, or to go to another system.

As the Cogo compiler is an adaptation of original IBM FORTRAN without format, (F0-002), which compiles programs to bulkier machine language than later processors, it was decided to attempt a hand compilation of the FORTRAN, using the FORTRAN subroutines for all arithmetic.

When this was accomplished, we found that we were still short of core, but were within striking range of fitting if we gave up parts of Cogo Basic, and rewrote other parts in machine language.

This created new problems, since the Cogo monitor was only designed to read in plug decks which do not overlay Cogo Basic. A simple modification to the Cogo Monitor allows the use of self-loading programs as Cogo plug decks.

With this completed, our program fit into memory, and we were able to proceed with debugging.

Since the elevations plug deck overlays part of Cogo Basic, it is necessary to do something to restore Cogo Basic prior to the use of any commands not in elevations. This is accomplished by the command RESTORE/BASIC which initiates the loading of a small deck of cards which puts back the part of Cogo Basic that elevations overlays.

MODIFICATION OF COGO FOR 1443 AND CARD OUTPUT

Since the IBM version of Cogo for a card system has typewriter output, making any short problem a long affair, we decided to give Cogo other output capabilities.

The restrictions which we placed on ourselves were that we wanted to use no more memory than was used for the typewriter-output Cogo, and that we did not want to have to recompile any of the plug decks in the system.

This required two steps. The first of these was to change the subroutines. This was accomplished by writing our own output routines and replacing the routines called PRTFX and PRTFL in the compiler with the new routines. This entails changes in only Cogo Load, and not in any plug decks.

The second change required was a change in that which is compiled for a PRINT statement. A program was written to seek out the compiled print statements in a plug deck, and change from the old sequence to our new sequence. The essential change is that on the typewriter the instruction for a new line is given at the start of the sequence, whereas for card or printer output, the print or punch must be given after everything is put in the output area.

The search and modify program, as it was written contains provision for fixed or floating point, unsubscripted or single-subscripted variables. It will not handle double-subscripts, but this presented no problems as none of the Cogo plug decks use double subscripting.

To modify a plug deck, the deck is loaded as if to use it in a problem. After resetting, the modify program is loaded. This program is set into operation, and continues until it runs out of core. It then check stops. (As the program was designed for limited use, and the stepping up core was done in many places, testing for the top of memory was felt to be unnecessary.) After pressing reset, the instructions on page 51 of the IBM Cogo writeup are to be followed, starting with step C.

This produces a system which will run any Cogo problem that will run on the typewriter system, and run it more quickly.

COGO DUMP/ON/CARDS

Since we found that often it is required to use points calculated in a Cogo problem run one day in a problem to be run at some other time, it was decided that we would add to the system the capability to punch the coordinate table in a form suitable for future input to the system.

This program was written in SPS and punches the points in the coordinate table which are defined. The format of the cards is that of a STORE command. The word STORE is in columns 1 through 5, the point number is in columns 21 and 22, etc. The problem number is punched in columns 77-80 for identification purposes.

The procedure is simple. The call command says

```
CALL                    END/OF/PROGRAM/DUMP
```

The command itself is

```
DUMP/ON/CARDS    N    J    ID
```

which says to dump from point N through point J with the number ID in columns 77-80

COGO ELEVATIONS ROUTINE

This program is for the purpose of finding the elevations of points with known coordinates. It is most useful for determining bridge deck elevations.

The program allows the user to specify a horizontal curve, if there is one, a vertical curve, if one exists, and different cross section slopes before transition, during superelevation and after superelevation, with the transitions handled by linear interpolation of slopes.

After calling the elevations plug deck, the first three commands issued must be, in order:

DEF/HORIZONTAL/CURVE

DEF/VERTICAL/CURVE

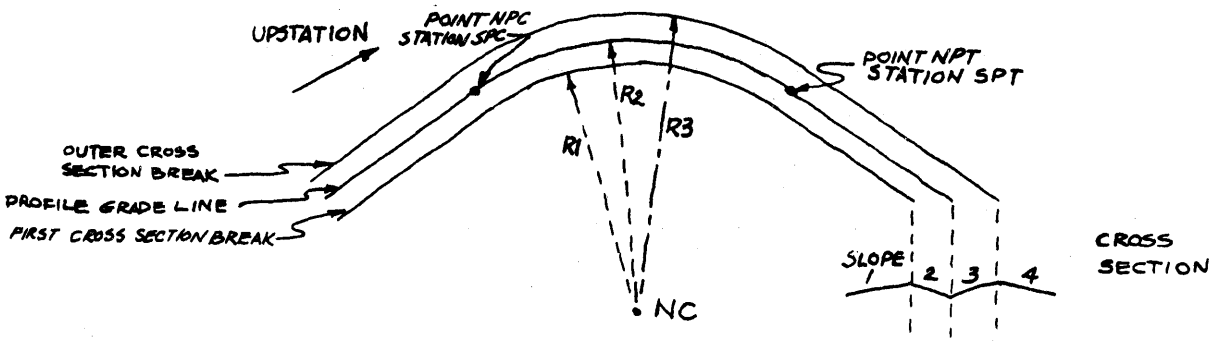
DEF/SUPERELEVATION

After these three setup commands have been issued, four other COGO commands may be used. The POINT/ELEV command gives the station, offset and elevation of the given point, while the KTH/PT/ELEV command divides the line between I and J into K parts and computes the station, offset and elevation of each point. The STA/OFFSET/ELEV command computes the elevation of a point with given station and offset. The RESTORE/BASIC command may be used to regain use of commands in Cogo Basic and must be used prior to any CALL statement after ELEVATIONS plug deck.

Note that while the elevations deck is in memory, the only Cogo commands which may be used are the above mentioned seven commands plus the STORE and PAUSE commands. Other commands may not be used even though they are usually considered to be in all plug decks. Comment cards may be used.

DEF/HORIZONTAL/CURVE NPC SPC NPT SPT NC R1 R2 R3 ELEVATIONS

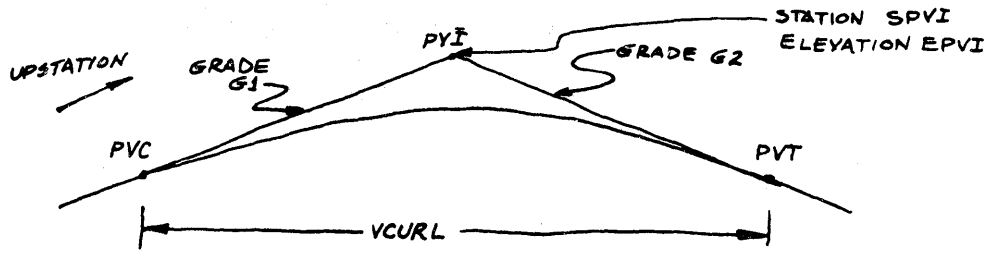
NPC Point number of PC of horizontal curve
SPC Station of PC
NPT Point number of PT of horizontal curve
SPT Station of PT
NC Point number of center of circle
R1 Radius from center of circle to first cross section break
R2 Radius from center of circle to profile grade line
R3 Radius from center of circle to outside cross section break



Note that cross-slopes are always specified starting with the side closest to NC. Also, the sign convention for these slopes is as follows: up and away from NC is positive.
If no horizontal curve exists then set the PC and the PT at opposite ends of the PGL. Set NC as some point on either side of PGL, not on it, remembering that cross slopes are specified starting closest to NC. Set R1 = 0. Set R2 equal to the distance from first cross-slope break to PGL. Set R3 equal to the distance from first cross-slope break to last cross-slope break. Note that R1 $\leq$ R2 $\leq$ R3. SPT must be greater than SPC.

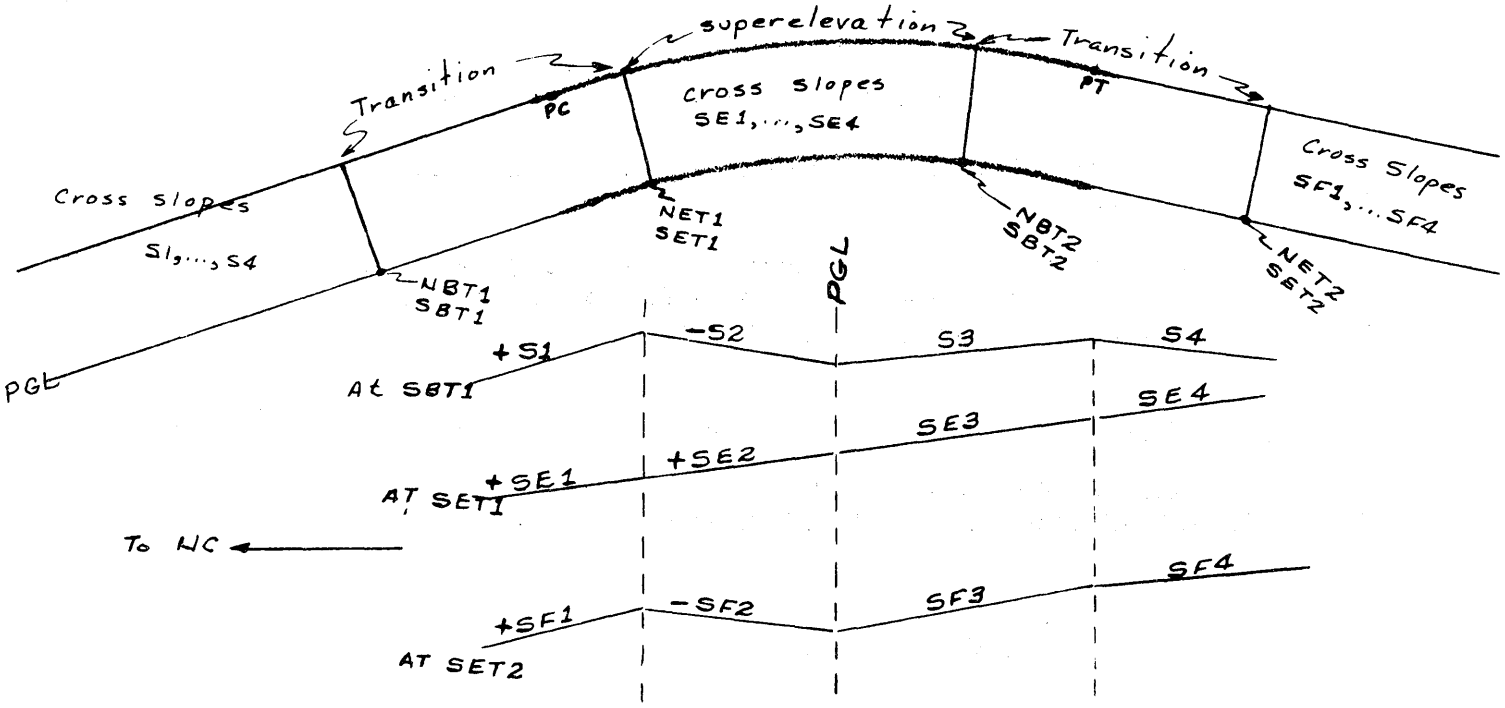
DEF/VERTICAL/CURVE NPVI SPVI EPVI VCURL G1 G2 ELEVATIONS

NPVI Point number of point of vertical intersection (PVI) of vertical curve
SPVI Station of PVI
EPVI Elevation of PVI
VCURL Vertical curve length
G1 Grade from PVC to PVI in ft per ft
G2 Grade from PVI to PVT in ft per ft



If there is no vertical curve, set the PVI as some point on the PGL having station SPVI and elevation EPVI. Set VCURL = 0.0. Set G1 = G2. Note that when this is done, an error message will be typed out but answers will be correct. The error message indicates only that VCURL = 0.0.

DEF/SUPERELEVATION	NBT1 SBT1 NET1 SET1 S1 S2 S3 S4 SE1 SE2 SE3 SE4 NBT2 SBT2 NET2 SF1 SF2 SF3 SF4
NBT1	Point number at start of first transition
SBT1	Station at NBT1
NET1	Point number at end of first transition
SET1	Station at NET1
S1,S2,S3, S4	Cross slopes at SBT1, starting with side closest to NC.
SE1,SE2,SE3,SE4	Cross slopes at SET1, starting with side closest to NC.
NBT2	Point number at start of second transition
SBT2	Station of NBT2.
NET2	Point number at end of second transition
SET2	Station of NET2.
SF1, SF2,SF3,SF4	Cross slopes at SET2, starting with side closest to NC.



If the cross section is constant, then set points for beginning and end of transition apart from each other and set SE1 = SF1 = S1, SE2 = SF2 = S2, etc.

If there is a two-phase transition into superelevation, use S1,...,S4 as starting slopes, SE1,...,SE4 as slopes between transitions, and SF1,...,Sf4 as slopes of full superelevation. Set NET1 = NBT2 and SET1 = SBT2.

POINT/ELEV N

N Number of point for which elevation is desired

Output: N STATION ELEVATION OFFSET

KTH/PT/ELEV K I J

K Number of intervals between I and J
I Initial point for which elevation is desired
J Final point for which elevation is desired

This command divides the line between I and J into K parts and gives elevations at each point.

Output:	DIST1	DIST2		
	I	STA(I)	ELEV(I)	OFFSET(I)
	⋮	⋮	⋮	⋮
	99	⋮	⋮	⋮
	⋮	⋮	⋮	⋮
	J	STA(J)	ELEV(J)	OFFSET(J)

Where DIST1 is the distance between any two consecutive points and DIST2 is the total distance from I to J.

Note that the coordinates of intermediate points are neither stored nor printed. If point number 99 is defined before entering the routine, it will be destroyed by the routine.

STA/OFFSET/ELEV STA OFF

STA STATION OF DESIRED POINT
OFF OFFSET OF DESIRED POINT FROM PGL

OUTPUT:

99 STA ELEV OFFSET

RESTORE/BASIC

This command initiates loading of a small deck of cards which enable one to use the commands of Cogo Basic again after using the elevations routine. This should follow the last computation command used after calling Elevations.

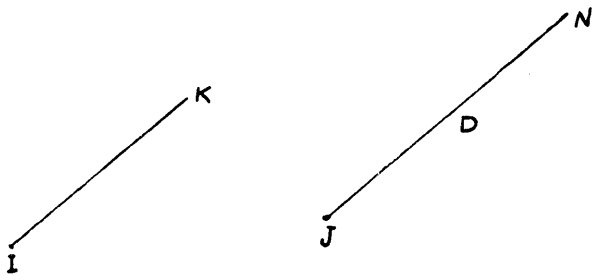
After this command is issued, any of the plug decks may be called.

PLUG DECK

LOCATE

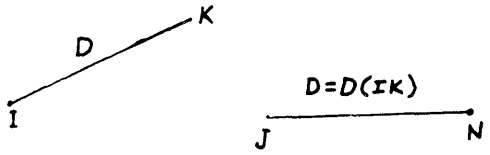
LOCATE/SAMEAZ J N D I K A

From J locate N at a distance D along an Azimuth the same as the Azimuth from I to K adjusted by an angle A. +A is clockwise
Output: Coord. of N.



LOCATE/DISTANCE J N I K A LOCATE

From J locate N along an Azimuth A at a distance equal to the distance from I to K.
Output: Coord. of N.



CARD DUMP

DUMP/ON/CARDS I J

Dump onto cards in format suitable for input, the point number and coordinates of each non-cleared point starting with point I and going up to point J
Output: STORE command for each point punched on cards

TWELVE DIGIT COGO

In order to answer the desire of some of our clients for more accuracy in geometry problems, we have written a program which is similar in input and output to Cogo, but which carries 12 digits instead of 8, as Cogo does.

The commands available in Twelve-Digit Cogo are:

	INTERSECTIONS PLUG	ALIGNMENT PLUG
STORE	YES	YES
PAUSE	YES	YES
LOCATE/AZIMUTH	YES	YES
INVERSE/AZIMUTH	YES	YES
LOCATE/SAMEAZ	YES	YES
LOCATE/DISTANCE	YES	YES
ARC/LINE/AZIMUTH	YES	NO
ARC/LINE/POINTS	YES	NO
ARC/ARC/INTERSECT	YES	NO
POINTS/INTERSECT	YES	NO
AZ/INTERSECT	YES	NO
REDEFINE	YES	YES
DUMP/ON/CARDS	YES	YES
*DIVIDE/LINE	YES	YES
GIRDER/LENGTHS	YES	YES
CLEAR	YES	YES
CALL	YES	YES
COUNT	YES	YES
ALIGNMENT	NO	YES

*This command differs from the Divide/Line in Cogo in that it references no stored curve, and gives no stationing or offsets.

The commands are written exactly as they would be for Cogo. Note, however, that there are only two plug decks, with more commands per plug deck, thus requiring fewer call statements.

By means of the Dump/on/Cards command, the coordinate table can be output in a form acceptable to either regular or 12-digit Cogo. This allows for the use of elevations, part of regular Cogo, with points established by 12-Digit Cogo.

NEW YORK STATE
DEPARTMENT OF PUBLIC WORKS

HAROLD M. GOTTHEIM
ASSOCIATE CIVIL ENGR. (ELEC.)
BUREAU OF ELECTRONIC DATA PROCESSING

1220 WASHINGTON AVENUE
ALBANY, NEW YORK 12226
GL 7-2458

Single Box Culvert

Program Number 2830

General

Given the clear span, clear height, height of fill, the thickness of the walls and slabs and the type of live loading, the program outputs the maximum moments and maximum required areas of steel at all the critical points on the culvert together with the distance from the point of zero moment to the outside face of the wall for the top and bottom slab. The location of the live load for each maximum condition, namely the distance from the centerline of the wall to the first wheel load, is also outputted together with miscellaneous information such as the required perimeter of steel in the top and bottom slab, the maximum soil pressure and the volume of concrete per foot of box. ($\frac{1}{2}$" is added to the top of the bottom slab)

The program does a unit moment distribution by placing a fixed end moment of 1 kip-ft. on the end of one member and distributing the moments. This is done for each of the eight member ends that can receive a fixed end moment. The 64 answers are saved for later use.

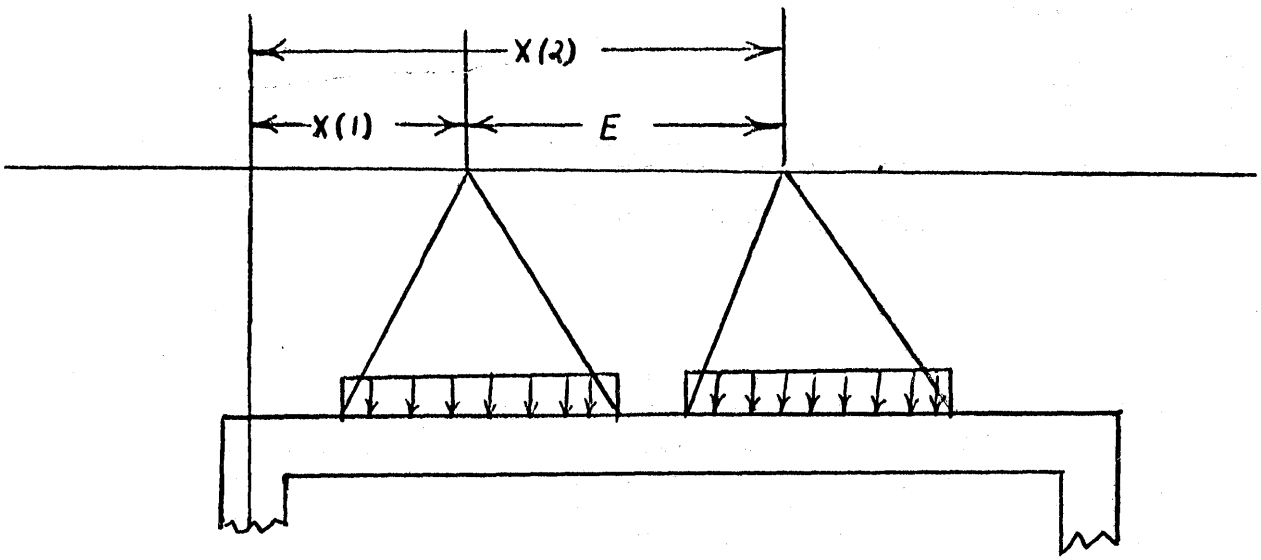
The impact coefficient, wheel loads and fixed end moments are found.

The maximum soil pressure is computed by $P/A \pm MC/I$. A two foot surcharge of soil is added for all cases containing live load. for side load

If the fill is greater than 8' and is also greater than the span, the live load is not considered. In all other cases, the live load is considered and is applied for the six cases as shown in the following diagram.

The wheel load is distributed over a distance of 1 3/4 times the fill longitudinally. When the fill is less than 2', the wheel load is distributed 6" (this is used to simplify computations) longitudinally. All wheel loads are distributed 5' laterally.

Fill	Impact
0---------1'	1.3
1' 1"-----2'	1.2
2' 1"-----2'11"	1.1
3' 0"-----	1.0



X (1) = 0, then 0.1 span, 0.2 span, 0.3 span, 0.4 span, 0.5 span
E = 14.0' for HS20-63T loading
E = 4.0' for military loading
X (1) = 200.' for the case of no live load

Now that all fixed end moments are known, the corner moments are found by using the unit distribution, the unbalanced shear is corrected, reactions are found and shear is checked. If shear exceeds the allowable, the slab is incremented until the shear is below the allowable. The slab moments and wall moments are computed for each tenth point and the largest positive and negative moments are saved. This is done for every live load case (if applicable) and the case of no live load and for each load type.

Cover = 2" except for bottom of top slab which is $1\frac{1}{2}$"
Bars of 1" diameter are assumed, therefor;

d for design = thickness -2" for bottom of top slab and top of bottom slab.
d for design = thickness $-2\frac{1}{2}$" for all other reinforcement.

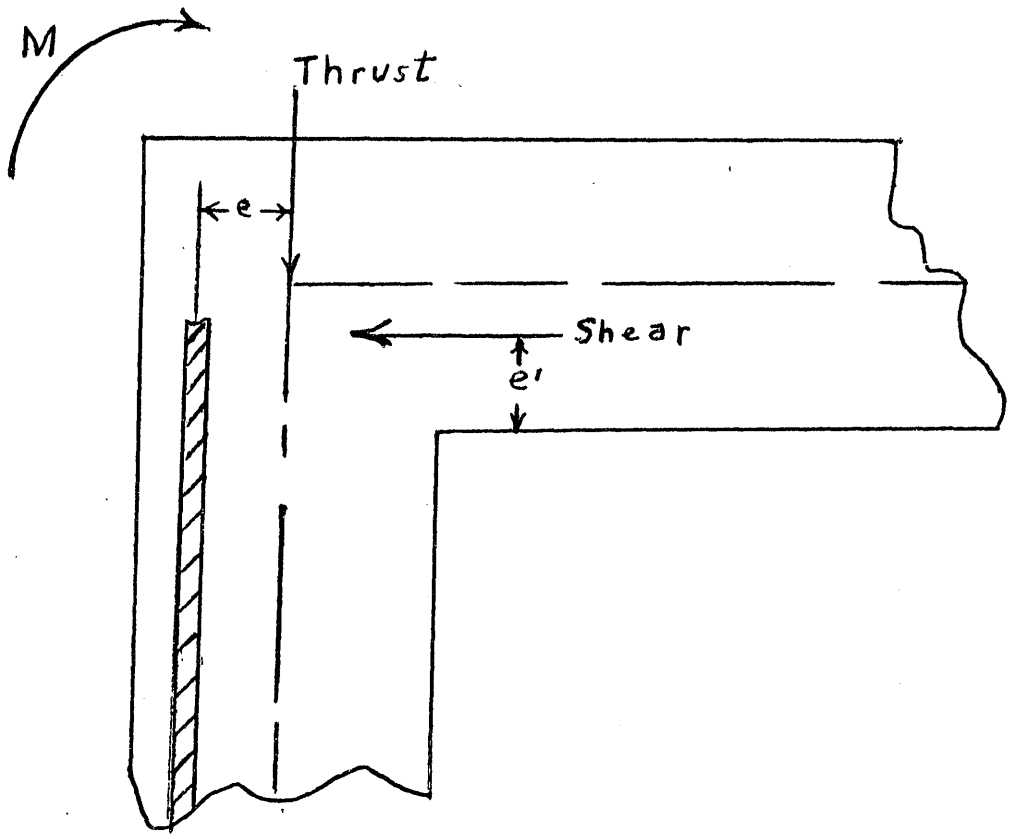
The moments labelled corner are the computed moments at the intersections of the wall and the slab centerlines. This corner moment is called M in the following paragraphs.

CRRM1, CRRM4, CRRM3, CRRM7 are moments which are found by using the following formula: (diagram on next page)

Moment = M - e'*SHEAR

The steel for these points is designed for combined bending and axial stress using the following formula:

$$A_s = \frac{\text{Moment (from above)}+e*\text{Thrust}}{f_s jd} - \frac{\text{Thrust}}{f_s}$$

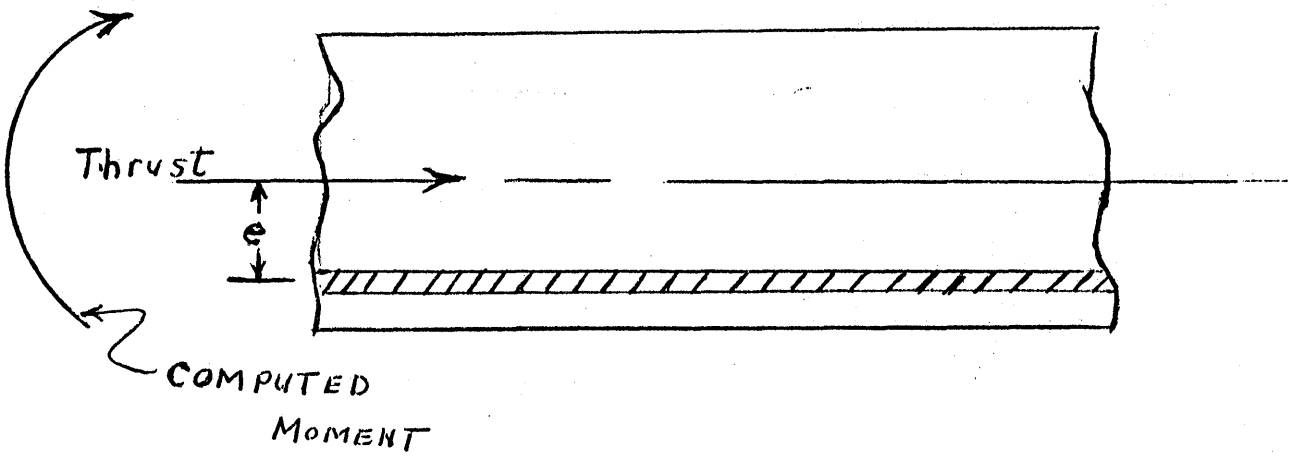


M = Computed corner moments
e = Distance from the point of applications of the thrust to the centerline of the tensile steel being designed
e' = 1/3 the thickness of the adjoining member

TOP1P, BOT1P, WAL1P, WAL1N are moments computed at the centerline of the member, (diagram on next page)

The steel for these points is designed for combined bending and axial stress using the following formula:

$$As = \frac{\text{Computed moment (TOP1P,BOT1P,etc)+e*Thrust}}{f_s jd} - \frac{\text{Thrust}}{f_s}$$



The culverts are designed in accordance with the procedures outlined in the A.A.S.H.O. Standard Specifications for Highway Bridges, Eighth Edition, 1961 and modified by the Standard Practices for Design, Detailing and Notes, State of New York, Department of Public Works, Division of Construction, Bridge Subdivision.

A FORTRAN listing of the program is attached to define engineering formulae, parameters, methods and Sequence of Operations.

Description of Typical Design Output

Items FS, FC, J, K, R, ERTH, etc. are constants of the program. They are always the same unless the designer chooses to have them changed. They can be changed for any special run or runs by contacting the Bureau of Electronic Data Processing or your EDP liason man.

The line starting with span, height, etc. are values the designer furnished for this program.

The values labelled allowable moments in concrete are the allowable moments at that section without exceeding the allowable concrete compressive stress. (for a balanced beam)

The moments and areas of steel are shown as an outline of the box. That is, the lines drawn on the typical output sheet (next page) can be assumed to represent the box itself. Values on the right half are moments and their mirror image on the left half gives the corresponding areas of steel at those points.

The controlling moments are printed out along with the value of X(1) and the type of loading that caused that maximum moment.

NOLL means no live load

MIL. means military loading

HS20 means HS20 loading

The next line gives manimum soil pressure; volume and required perimeters of steel for bond. The weight of the walls is included in the maximum soil pressure. A 1/2" addition to top of bottom slab is also figured in the volume and soil pressure.

Point of zero moment and maximum shear are shown last and are self-explanatory.

INPUT IDENTIFICATION

Constant Card (Last card in object deck)

Symbol	Field	Explanation
FS	xxxxx.xx	Allowable tension steel stress - P.S.I.
FC	xxxx.xx	Allowable stress in concrete - P.S.I.
CJ	xx.xxx	j
CK	xx.xxx	k
CR	xxx.x	R or K
ERTH	xxx.xx	Weight of Earth - #/ft^3
CONC	xxx.xx	Weight of Concrete - #/ft^3
UTOP	xxx.x	Allowable bond stress in top bars - P.S.I.
V	xxx.x	Allowable vertical shear - P.S.I.
FPR	xx.x	Equivalent Fluid Earth Pressure #/ft^3
SAM	x.x	That part of wt. of walls to be used in design (0.0 minimum and 1.0 maximum)
SAM1	x.x	That part of side load to be used to reduce positive moment in slabs (0.0 minimum and 1.0 maximum)
ALANE	x.x	Number of lanes, (Either 1.0 or 2.0) 2.0 if lanes are to overlap transversely.
BOTAD	x.x	The number of inches by which the bottom slab exceeds the top slab.

Program Card

SPAN	xxxxx.xx	Clear span - ft.
HT	xxxx.xx	Clear height - ft.
T_1	xx.xxx	Top slab thickness - inches
FILL	xx.xxx	Height of fill - ft.
TIPE	xxx.x	Type of loading
WALL	xxx.xx	Wall thickness - inches

Card No. 1, 2, 3, 4, 5 etc. are similar

One program card for each design.

Double Box Culvert

Program Number 2831

General

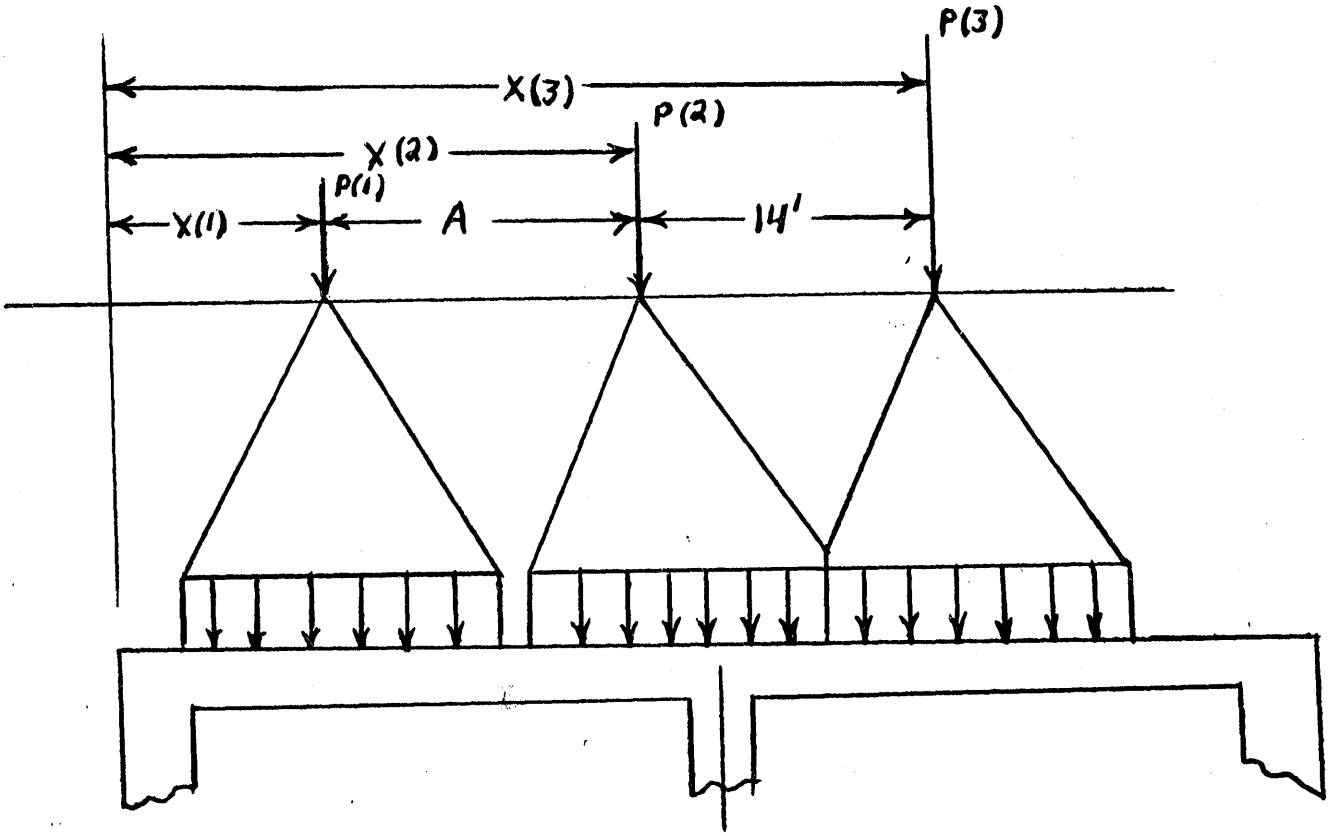
Given the clear span, clear height, the thickness of slabs and walls, the type of loading and the height of fill, the program outputs the maximum moments and maximum required areas of steel at all the critical points on the culvert together with the distance from the point of zero moment to the center of the wall for the top and bottom slab. The location of the live load for each maximum condition, namely the distance from the centerline of the wall to the first wheel load, is also outputted together with miscellaneous information such as the required perimeter of steel in the top and bottom slab, the maximum soil pressure and the volume of concrete per foot of box.

Typical Design Output

The program does a unit moment distribution by placing a 1 kip-ft. fixed end moment on each of the 14 member-ends that can receive a fixed end moment, and doing a moment distribution for each case. The answers are stored for future use.

The fixed end moments due to dead load, and side load are computed and saved. (A 2 ft. live load surcharge is added to the fill for the conditions which include live load). The fixed end moments due to shrinkage, temperature fall and temperature rise are computed and saved. (These can be made zero by changing the value of the shrinkage coefficient to zero on the constant card. The corner moments caused by temperature change, shrinkage and side load are computed and saved.

The live load impact coefficient is found. If the fill is greater than 8' and also greater than the sum of the two spans, the live load is not considered. In all other cases, the live load is applied as shown in the diagram below.



For HS20-63T loading A = 14', P(1) and P(2) = 3200 x (Impact coeff.), P(3) = 800 (Impact coeff.).

For Military loading A = 4', P(1) and P(2) = 3200 x (Impact coeff.), P(3) = 0.

X(1) = 0, then 0.1 span, 0.2 span.....1.0 span. The last condition is always X(1) = 200 for the case of no live load.

The wheel load is distributed over a distance of 1 3/4 time the fill longitudinally. When the fill is less than 2', the wheel load is distributed 6" longitudinally. (This is done for ease of computations). All wheel loads are distributed 5' laterally.

By use of the unit distribution, the balanced corner moments for the combined effect of live and dead load are computed for each case of live load. Shear is checked in both top and bottom slabs and the thicknesses of the slabs are incremented as needed to satisfy shear. If either top or bottom slab is incremented, the program returns to the beginning and starts over with the corrected slab thicknesses. If shear does not exceed the allowable stress, the program continues.

The program now computes the moment at each 1/10 point to find the critical points. As the program goes through each loading condition, the biggest positive or negative moment is saved (whichever is applicable). The effects of temperature, shrinkage and side load are now added if they increase the moments at the sections we are interested in.

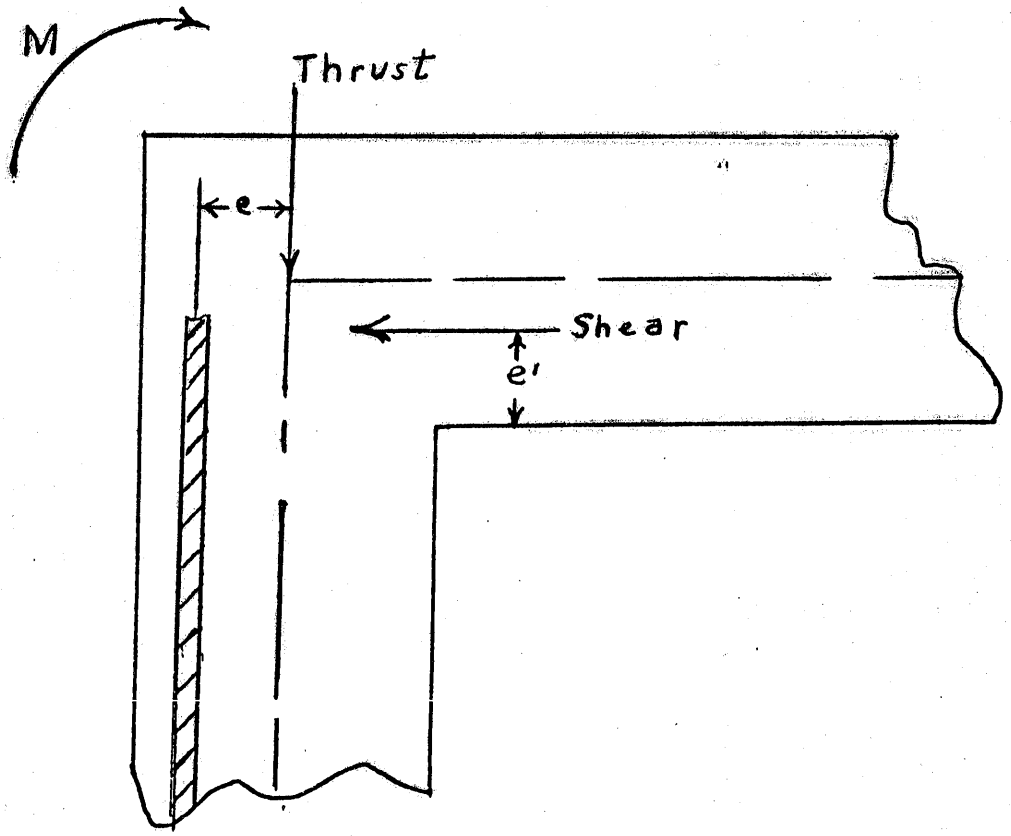
The moments labelled corner are the computed moments at the intersections of the wall and the slab centerlines. This corner moment is called M in the following paragraphs.

CRRM1, CRRM2, CRRM3, CRRM4, CRRM7, CRRM8 are moments which are found by using the following formula: (diagram on next page)

$$\text{Moment} = M - e' * \text{SHEAR}$$

The steel for these points is designed for combined bending and axial stress using the following formula:

$$A_s = \frac{\text{Moment (from above)} + e * \text{Thrust}}{f_s jd} - \frac{\text{Thrust}}{f_s}$$

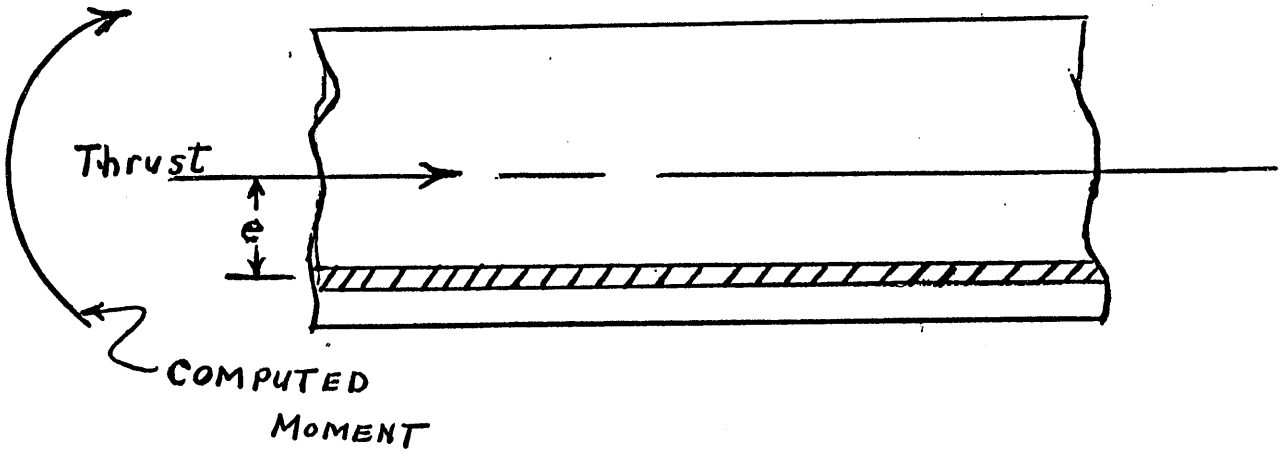


M = Computed corner moments
e = Distance from the point of applications of the thrust to the centerline of the tensile steel being designed
e' = 1/3 the thickness of the adjoining member

TOP1P, BOT1P, WAL1P, WAL1N are moments computed at the centerline of the member, (diagram on next page)

The steel for these points is designed for combined bending and axial stress using the following formula:

$$As = \frac{\text{Computed moment (TOP1P,BOT1P,etc)}+e*\text{Thrust}}{f_s jd} - \frac{\text{Thrust}}{f_s}$$



Fill	Impact
0----------1'	1.3
1' 1"-----2'	1.2
2' 1"-----2'11"	1.1
3' 0"-----	1.0

Cover = $2\frac{1}{2}$" except for bottom of top slab which is $1\frac{1}{2}$".

d for design = thickness - (cover + $\frac{1}{2}$)

The culverts are designed in accordance with the procedures outlined in the A.A.S.H.O. Standard Specifications for Highway Bridges, Eighth Edition, 1961 and modified by the Standard Practices for Design, Detailing and Notes, State of New York, Department of Public Works, Division of Construction, Bridge Subdivision.

A FORTRAN listing of the program is attached to define engineering formulae, parameters, methods and Sequence of Operations.

INPUT IDENTIFICATION

Constant Card No. 1

Symbol	Field	Explanation
FS	xxxxx.x	Allowable steel stress - p.s.i.
CR	xxxx.	R or K
UBOT	xxx.x	Allowable bond stress in bottom bars - p.s.i.
UTOP	xxx.x	Allowable bond stress in top bars - p.s.i.
SAM	xxx.x	That part of wt. of walls to be used in design (0.0 minimum to 1.0 maximum)
FPR	xxx.x	Equivalent fluid earth pressure - p.s.i.
V	xxx.x	Allowable vertical shear - p.s.i.
AN	xxx.x	n
CJ	.xxx	j
ANCOF	x.x	1.0 if using (n-1), 2.0 if using (2n-1)
ALANE	x.x	Number of lanes that can overlap transversely (1.0 or 2.0)

Constant Card No. 2

FSC	xxxxx.x	Allowable stress, compr. steel - p.s.i.
FC	xxxx.	Allowable stress in concrete - p.s.i.
ERTH	xxx.x	Weight of earth - #/ft.^{3}
CONC	xxx.x	Weight of concrete - #/ft.^{3}
SHCOF	xxx.x	Shrinkage coefficient for concrete In./In. x 10^6
BOTAD	xxx.x	The number of inches by which the bottom slab exceeds the top slab
TFALL	xxx.x	Temperature fall - °F
TRISE	xxx.x	Temperature rise - °F
CK	.xxx	k
ASCOF	x.x	That part of positive slab steel which is carried over center support and is to be used as compressive steel. (0.0 minimum to 1.0 maximum).
SAM1	x.x	That part of side load to be used to reduce positive moment in slab. (minimum is 0., maximum is 1.)

Program Card #1

OSPAN	xxxxx.x	Clear span in feet
TIPE	xxxx.	Type of loading used
OT1	xxx.x	Top slab thickness in inches
CEWAL	xxx.x	Center wall thickness in inches
WALL	xxx.x	Outside wall thickness in inches
OHT	xxx.x	Clear Height in feet
OFILL	xxx.x	Height of fill in feet

EDP (5/65)

MADE BY_____ DATE_____

CHECKED BY_____ DATE_____

BACK CHECKED_____ DATE_____

CALCULATION FOR_____

N.Y.S.D.P.W.

DESIGN OF DOUBLE
CONCRETE BOX CULVERT
INPUT FORM
2831.02

SQUAD NO._____

JOB NO._____

SHEET_____ OF_____

SWITCH SETTINGS: PARITY & 1/0 - STOP.....OVERFLOW - PROGRAM.....1, 2, 3 & 4 - OFF

CONSTANT CARD NO. 1 - NEXT TO LAST CARD IN OBJECT DECK - DO NOT PUNCH

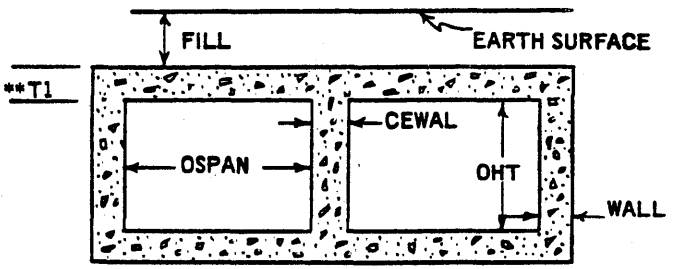
1 7 FS	8 12 CR	13 17 UBOT	18 22 UTOP	23 27 SAM	28 32 FPR	33 37 V	38 42 AN	43 46 CJ	47 49 ANCOF	50 52 ALANE
P.S.I.		P.S.I.	P.S.I.	0. to 1..	P.C.F.	P.S.I.	Es/Ec			
2000000.10	0.11917.	300001.00	118001.00	000001.00	003001.00	019001.00	011001.00	.81750	11.10	11.10

CONSTANT CARD NO. 2 - LAST CARD IN OBJECT DECK - DO NOT PUNCH

1 7 FSC	8 12 FC	13 17 ERTH	18 22 CONC	23 27 SHCOF	28 32 BOTAD	33 37 T FALL	38 42 T RISE	43 46 CK	47 49 ASCOF	50 52 SAM 1
P.S.I.	P.S.I.	P.C.F.	P.C.F.		INCHES	° F	° F		0. to 1.	0.to 1.
1600000.10	12000.	12001.00	15001.00	000001.00	000001.00	000001.00	000001.00	.13750	11.10	00.10

1 7 OSPAN	8 12 TIPE	13 17 OT1	18 22 CEWAL	23 27 WALL	28 32 OHT	33 37 OFILL
CLEAR SPAN	TYPE OF LOADING	TOP SLAB THICKNESS	CENTER WALL THICKNESS	OUTER WALL THICKNESS	CLEAR HT	HEIGHT OF FILL
FT	*	INCHES	INCHES	INCHES	FT.	FT

FS	All. stress-tension steel
FSC	All. stress-compr. steel
CR	R or K
UBOT	Allowable bond stress in bottom bars
UTOP	Allowable bond stress in top bars
SAM	% of wt. of walls used in design
FPR	Equiv. fluid earth pressure
V	Allowable shear
AN	n (ES/EC)
CJ	j
ANCOF	1. if using n-1, 2 if using 2n-1
ALANE	2. if lanes overlap transversely, otherwise 1.
FC	Allowable stress in concrete
ERTH	Wt. of earth
CONC	Wt. of concrete
SHCOF	Concrete shrinkage coeff. * 10^6 (usually 0.-or 200.)
SAM1	% of side load to be used to reduce positive moment in slab
BOTAD	The number of inches by which the bottom slab t exceeds the top slab t
TFALL	Temperature Fall
TRISE	Temperature Rise
CK	k
ASCOF	% of positive steel carried over center support.



SPAN = skew span for 0° to 20° skew.
= right angle span for 20° and over.

* TIPE=1. HS20-63T
TIPE=2. HS20-63T or military loading, whichever is greater.

** Minimum dimension desired.

DESIGN OF DOUBLE BOX CULVERT

CONSTANTS

20000.0	197.0	300.0	180.0	.0	38.0	90.0	10.0	.875	1.0	1.0	
16000.0	1200.0	120.0	150.0	.0	.0	.0	.0	.371	1.0	.0	

INPUT

OSPAN	OHT	OT1	CEWAL	WALL	TIPE	OFILL
23.2	10.0	18.0	12.0	18.0	2.0	3.0

	ASN1	ASP1	ASN7	X(1)= 17.1 CRRM2	X(1)= 12.2 TOP1P	CRRM1	X(1)= 7.3 CORNER
	-.96	1.42	-2.04	-46.20	33.16	-23.51	-28.27

ASN2	CRRM4
-.93	-26.76

ASN3	ASP2	X(1)= 24.4 WAL1P	X(1)= 12.2 WALIN
-.85	.00	.00	-24.07

ASN4	CRRM3
-1.09	-27.94

	ASN5	ASP3	ASN6	X(1)= 17.1 CRRM8	X(1)= 2.4 BOT1P	CRRM7	X(1)= 17.1 CORNER
	-1.10	1.56	-2.58	-58.52	36.48	-27.01	-29.71

MAX. SOIL PRESS.= 1.2 K.S.F. VOL, 7.2 C.Y.

DIST. FROM CENTER OF WALL TO PT. OF ZERO MOM.		MINIMUM PERIMETER FOR REQ. STEEL		MAX.
	OUTER WALL	CENTER WALL	OUTER WALL	CENTER WALL	SHEAR
TOP SLAB	3.9 FT.	6.3 FT.	3.7 IN.	4.3 IN.	67.6
BOT SLAB	12.2 FT.	12.2 FT.	1.1 IN.	2.1 IN.	57.2

COMPUTED DIMENSIONS

T1= 18.0 T2= 18.0 FILL= 3.0

MORE/LESS - CPM AND PERT FOR DISK AND PRINTER

Authors: Melvin DeSpain and Alan D. Johnson

National Aeronautics and Space Administration
Plum Brook Station
Sandusky, Ohio

Introduction

PERT[1] and CPM[2], perhaps because of their origins and their first applications, have generally been thought of as being limited to large, high-speed, computers; although programs for smaller machines with lower through-put speeds have been written. Core size limitations have generally made these small machine programs less efficient than could be desired from the standpoints of the number of jobs that can be handled for a given project and the number of data input and output passes that must be made. The addition of random access disk files and the medium-speed on-line printer now allow the formulation of better programs for CPM and PERT. Throughout the following, PERT and CPM are used synonymously. The two techniques, although similar in many ways, are not exactly the same. The really basic differences are two in number; the first being that CPM provides as its major output the scheduled completion date for the project, determined from the job input data, whereas PERT, with a fixed and arbitrarily assigned project completion date, reports how far the project is ahead or behind this date. The second major difference, now fast disappearing as various versions of PERT are formulated, reports the status of the nodes of the arrow diagram, whereas CPM reports upon the individual jobs and activities. This type of report is also available under many PERT systems and, in fact, in our organization is the one finding greater use. Since the first steps of both phases of PERT and CPM are essentially the same, with the differences occurring in the output format, they are treated synonymously.

The work described in this paper was done to provide CPM capability for NASA Plum Brook Station, using the 1620 system. The requirement here is to follow small to medium size construction and research equipment installation projects. The very large projects are programmed for the high-speed computer installation at our Cleveland headquarters.

The programs for this system have been written for the 1620 Mod. I with a 20K core, a 1622 card-read punch, a 1311 disk file, and a 1443 printer. The Monitor I system is used, as well as the special features (indirect addressing, TNS, TNF, etc.).

Note 1. Project Evaluation and Review Technique
Note 2. Critical Path Method

System Requirements

In order to develop a useful system, a review of the existing 1620 programs was made. Their difficulties and limitations were examined in the light of the requirements given below. The first general requirement was that any system developed must place the least strain possible upon the engineering manager. This, in turn, means that minimum limitations must be placed upon the preparation of the original arrow diagram. It also requires that the preparation of input material must be easy and straight-forward. Another prime requirement is that the output shall be both current and easily understood. It has been adequately demonstrated by people working in the field of PERT and CPM that failure to meet the above requirements reduces the work performed to a mere exercise.

Other requirements perhaps peculiar to our open shop operation are that the program systems shall require only a minimum understanding of console procedures. This involves making the system as nearly self-protecting and fail-safe as possible within the other restrictions imposed. Also, because of the open shop operation, through-put becomes important since it may be engineering personnel standing by during the running of the system rather than clerical. In addition, there are the obvious machine costs to be considered.

Lastly, the requirement for system flexibility was imposed. This meant the ability to produce either PERT or CPM outputs, the ability to handle input durations either in working days or actual calendar dates, and a wide selection of output formats as may be required by the job being run.

System Organization

Using the Disk Utility Program of Monitor, the various programs that make up the system can be stored with program names listed in the Equivalence table. Since there are a large number of programs involved, the use of this more-or-less permanent storage may be limited and, for that reason, a system of negative DIM numbers covering the program storage in the disk working cylinders was developed. The presence of a special loader in core as the object programs are loaded by the card reader will cause loading in the working cylinders; otherwise, normal *DLOAD action of the Monitor Disk Utility Program takes place.

Once the programs have been loaded on the disk, a starting program is run that builds a table of available programs by executing a search of the Equivalence table and the special negative DIM tables. If a given program is found in both places, a choice as to which one to use is allowed. This feature allows

modified versions of a permanently loaded program to be used, which is especially valuable during program development phases. In addition to the construction of the program table, subprograms and error messages are placed for easy availability and the company organization name entered. This program exits to a record analyzer (RCANYL) routine that is the tie between the various programs of the system.

When a particular job is finished, a cleanup program is called to core to clear the program prerequisite table and prepare for the next job. If the last job has been finished, this program will destroy any reference to the programs in working storage and exit to Monitor I Supervisor.

System Programs

As mentioned above, the system concept has been used for this work. This means that the programs are called as required, by the use of a supervisor type routine. Each program when called checks a table to see whether the required prerequisite program(s) have been run. Listed below, under the assigned Equivalence name, are each of the programs developed thus far:

PHASEA - This program stores the project name and reads in the job cards. In the process, it looks for blank fields, checks for more than one start or finish event, and looks for arrow diagram loops. To make use of the following program, the input deck is sorted so that all jobs having a common origin (I) are read in, one after the other. In order to provide a semblance of order (not required), they may be sub-ordered by the J number. With the input cards so ordered and under switch option, a table of addresses is built on the disk for use of follow-on programs. A J node frequency table is also built and stored. The raw data from the job cards is stored with one job per disk sector.

ORDRIJ - This is the key to rapid execution of the two programs following. Assuming properly assembled data from PHASEA, a table of job addresses in order of calculation for a single data pass is built and stored on the disk. Under switch option, an ordered list of jobs may be outputted on the printer and an ordered deck outputted from the card punch. Use is made of the J node frequency table in preparing the ordered table.

PHASEB - Using the general methods developed by Sauer in his MISS-LESS program, a table of I values is built in core and upon completion is placed on the disk. This program, under option, uses the table developed under ORDRIJ, making only one pass through the data. Without this option, as many passes are made as are required to complete the evaluation, as is done in the MISS-LESS program. It can be shown for the worst case that, without ordering, the number of passes through the data is equal to the number of jobs plus one. Ordering, then, becomes more important the larger the job to be done and the more random is the node number assignment.

PHASEC - Similar to PHASEB above, except that the J values are calculated and stored each to a data sector. Again, the option to use the table developed by ORDRIJ is available with the same time savings. In this case, the table of ordered addresses is run from rear to front.

CPCALC - Bringing in the I table produced under PHASEB and taking each job in turn, this program calculates earliest and latest start, earliest and latest finish, total float and free float and, in addition, marks each job data sector as to whether the job is critical (total float equal to zero) or not.

CRTPAT - Up to this point, the calculated data is not significantly different from that available upon the completion of MISS-LESS; that is, individual jobs have been marked as to whether they fall on some critical path. This program searches out the critical path or paths and builds and stores address strings for each. As an option when multiple paths are discovered, the duration of the second critical event can be reduced by one unit and the program call made to PHASEB. Through the multiple passes, a single critical path can be developed. The jobs with changed durations are marked for future output uses.

BILDAY - This is the first of the dating programs. This one builds a ten year calendar, complete with holidays, on disk storage. Two versions of this program have been developed. The first is self-contained, in that the starting date and holidays are part of the source program. The second version, with a call to the card reader for data, will build any ten year calendar with only the holidays requested through card input. A unique holiday occurring only once can be entered if desired.

BILDAT - This program builds a table of working days in core, starting with the project starting date. Work week length can be selected and the inclusion of holidays in the work week, if desired, is provided for. A four digit date representation is used. This program can also provide a six digit representation. A table of working days for each month is also built.

LSTEVN - This, the first of three output programs, will, under two options, either list all jobs, making note of those that are critical, or list only the critical jobs. Output, of course, is on the printer.

LSTPAT - Two output formats are provided by this program. One gives a parallel presentation of the jobs that make up each critical path. The second lists the critical jobs with their data for each critical path. Date or day presentation may be specified.

SEEPAT - This program represents the first attempt to visualize a critical path on a real time basis. In addition to plotting the path itself, the non-critical jobs leading to the successful attainment of any event are also shown. The first half of the printer sheet is given over to the display of the critical path, with the abscissa being a uniform time scale in working days. The second half of the sheet lists the data for those critical and non-critical events referenced on the diagram above. There is little question that this program will see many revisions resulting from feedback from the field.

Tentative Specifications

The input data card format is the same as that for MISS-LESS, except that the allowable job name cannot exceed twenty-six (26) alphameric characters. The only card output so far (ORDRIJ) is the same as for data input except for a sequence number appearing in columns 76 through 80.

Nodes may be numbered 1 through 3000, inclusive, and may be in any order relative to their appearance on the arrow diagram. The total number of jobs, then, is one less, or 2999.

Job durations are loaded to four digits and, for use with the dating programs, must be in days. Projects are thereby limited to slightly over 27 years, if dating is not used. With dating, the table in core covers 1499 working days or approximately 4 to 6 years, depending upon the length of the work week.

Money is limited to five digits for each job and little use is made of this data at the present time; however, some future programs now being planned will make more extensive use of this data.

Status

All the programs specifically mentioned in this paper have been written and debugged, with the exception of SEEPAT which is seeing much minor modification to improve upon the output format. As of the present writing, several sets of input cards have been used, but the largest project has involved little over 200 jobs. In addition, the input data was clean. The "idiot" tests are just starting. Since the system is to be used on an open shop basis by non-computer-trained personnel, it must be assumed that every possible thing that can be done wrong will be done wrong. The method of approach is for experienced people to try all of the mistakes they can think of, modifying the program so as to be fail-safe for each mistake. Next, the system is released to the field, with untrained people doing the work. Experienced personnel will be available to monitor the operation and make note of any other program deficiencies. Plans are being made to release the system to a few interested users so that the programs can be developed to be fully operational at the earliest possible date. It is intended that the system will be tendered to the Users Group Library if there is sufficient interest.

Future Plans

The system described provides for a total of 25 programs, leaving considerable room for expansion. A number of programs suggest themselves in the area of visual output. Another area of interest is job progress input (per cent completion) to handle the situation of very long jobs on the critical path. By far the most interesting is the problem of cost versus time, especially when dealing with construction projects. The question to be answered in this case is: Where can I most effectively spend some extra dollars to shorten the over-all project length? The limit of 3000 events is strictly one of core size and it is quite feasible to modify the program for larger projects. It is also possible that certain of the outputs that go to the printer could be modified for the card punch, making listing on a 407 or other similar equipment possible. Naturally, it is not the intention of the authors to address themselves to the problems involving other machine configurations.

Conclusion

By now, it is quite obvious that this paper is by nature a progress report and that it is unlikely that this system of programs will ever be fully complete since new ideas, techniques, and requirements constantly appear. Work thus far has, however, demonstrated that the addition of disk storage and line printer as peripheral equipment to the 1620 CPU makes possible, and practical, CPM and PERT programs for projects of medium size. It has also been shown that the establishment of an order for calculation very significantly reduces the time necessary for job time calculations.

MINUTES OF THE CHEMICAL ENGINEERING TEAM MEETING

Miami, Florida May 10, 1965

The Chemical Engineering Team met with the following in attendance:

A. H. Best
H. Gelsi
J. L. Jones
T. Korelitz
B. MacMullin
C. S. Schrodel
J. E. Wages

There were no papers to be presented, and so the meeting centered about a group discussion of applications and problems. Topics of interest were process design, process control, equipment design, and optimization. The discussion of process control centered about the need of an adequate process model and the difficulties of obtaining such a model.

Respectfully submitted,

C. S. Schrodel

C. S. Schrodel, Chairman

CSS:bah

FITTING STRAIGHT LINES TO X-Y DATA WHEN BOTH VARIABLES ARE SUBJECT TO ERROR

L. D. Y. Ong and F. K. Durkan
Health and Safety Laboratory
U. S. Atomic Energy Commission
New York, New York

Application of the least squares criterion for fitting a straight line to X-Y data is common knowledge and one of the most extensively-used statistical techniques. However, many analysts fail to fully appreciate the importance of an assumption stipulated by the method - that the independent variable, x, be either controlled or free from error.

It appears, surprisingly, that few analysts are familiar with possible alternate methods that might be applied to cases where both x and the dependent variable, y, are subject to error. This discussion attempts to acquaint the audience with this little-known and controversial area of statistics. We shall first discuss the general statistical problem and then consider in detail three methods of solution that are available.

In considering the relation between two physical quantities, it is usual to think of one variable as being the causal variable and to describe it as the independent variable, the other variable being dependent on it. In the statistical sense, there is no implication that the independent variable is causal. The values of the independent variable may have been fixed or selected in any manner. They need not be randomly distributed. When sampling is random with respect to both variables, either variable may be regarded as independent for prediction purposes.

This leads us to distinguish between the linear <u>regression</u> relation and the linear <u>functional</u> relation. The regression relation expresses the "expected" value of the dependent variable in terms of the "observed" value of the independent variable. The functional relation describes the "expected" value of the dependent variable in terms of the "expected" value of the independent variable. The essential distinction between the two lines then, is that the regression line refers to "observed" values of the independent variable, the functional line to "true" values.

The functional relation is required for the statement of laws in the empirical sciences which would hold if no errors existed. The functional relationship and the regression line are the same if, and only if, the independent variable is not in error. For example, the hypothesis of the existence of density is that $M = \rho V$ where ρ is the density and M and V represent mass and volume respectively. By the functional relationship we are solving for ρ. By the regression relation we are predicting "true" masses from "observed" volumes. This is possible since it is known that the mean of the "true" masses is linearly related to the "observed" volumes and the slope of this line is the regression coefficient of mass on volume. Hence the regression relation continues to have a meaning if both the variables are in error: it describes the relationship between the mean value of one variable and the other variable, and is thus a legitimate technique to use in the problem of prediction. Using the functional relation for purposes of prediction when both variables are subject to error results in predicted values that are biased.

Thus, not only do linear functional relations differ from regression relations in general, but they also have different applications. A theory may specify some relation among the "underlying" or "expected" values of certain variables. The functional relation would be of interest then, to determine whether the data support the specified form of relationship, as well as to estimate the parameters of the relationship or to check the correspondence of given parametric values with the observations.

The regression relations are based on the variation in both the "true" values and the random errors to which the observations are subject, while the functional relation is based on the variation in the "true values" alone. The functional relation is, therefore, relevant only to a study of how the "true" values of both variables are affected by some extraneous variable or variables, i.e., the relationship shows what elements of the system are invariant under changes in conditions. For example, in calibration experiments we are concerned with the "underlying" or linear functional relation existing between the results given by two instruments,

persisting through changes in conditions and regardless of the random errors to which the results may be subject. Generally, it is required that one instrument be capable of replacing the other under a wide range of conditions. Clearly, the greater the range of changes in conditions of the variables to be measured, the less the relative contribution of experimental error to the total variation of results, and the more closely the two regression equations (x on y, and y on x) will approach the functional relation.

When the independent variable is free of error, or "controlled" by experimental technique the variable may be dealt with as though it were errorless, since its "observed" and "expected" values coincide. For this case then, the regression relation is the same as the functional relation, and both may be estimated by the method of least squares. However, when the independent variable is subject to error or "uncontrolled", the coefficient of the regression line obtained by least squares is useless for examining the correspondence of data with a theoretical relationship since its value is affected by the magnitude of the error in the independent variable. Thus its slope will, on the average, be smaller than the true slope.

Consequently, if both variables are subject to errors, the problem of finding the best values of the empirical constants is more complicated. A common feature of most solutions proposed is the requirement that certain "a priori" assumptions (independent of the observations) regarding the standard deviations of the errors in X and in Y or at least, the ratio of the two error variances, be known. It is the investigation of three possible techniques for determining the functional relation when <u>none</u> of these three quantities are known accurately, that occupies the latter part of this talk.

Scarborough, Wald, and Bartlett, all propose a solution to the common case where the observations have equal weight, i.e., where both variables are subject to uncorrelated errors of the same order of magnitude.

Historically, the two regression lines (x on y and y on x) obtained by least squares have been termed the elementary regression lines. These are taken by most investigators as

fixing limits between which the best line required must lie. It is often maintained, further, that the best line is such that the mean-square deviation of the observations, taken perpendicularly to the line, is least. This line, which passes through the point of intersection of and lies in the acute angle between the elementary regressions is variously called the orthogonal regression line or the mutual regression line. Such a line is constructed by what we term the "Scarborough method" because of his excellent presentation of the method originated by Adcock, Pearson, and others.

The "Scarborough method" (1958) considers the line which best fits the observed points as that which minimizes the sum of the squares of the perpendicular distances from the points to the line.

The derivation of the method is as follows: The equation of any straight line may be written in the form

$$ax + by + 1 = 0. \tag{1}$$

This symmetrical form being used because both x and y are assumed equally subject to error. The sum of the squares of the perpendicular distances from the points (x_1, y_1), (x_2, y_2), etc. to the line is therefore

$$F\ (a,b) = d^2 = \frac{1}{a^2 + b^2} \left[(ax_1 + by_1 + 1)^2 + (ax_2 + by_2 + 1)^2 + \ldots + (ax_n + by_n + 1)^2 \right] \tag{2}$$

and since this is to be a minimum, its partial derivatives with respect to a and b must each be zero.

Taking the partial derivative of Equation (2) with respect to a, multiplying by a, then taking the partial derivative of Equation (2) with respect to b, multiplying by b, adding the results and simplifying, yields

$$a \frac{\partial F}{\partial a} + b \frac{\partial F}{\partial b} = - \frac{2}{a^2 + b^2} \left[a \Sigma x + b \Sigma Y + n \right] \tag{3}$$

But since $\frac{\partial F}{\partial a} = 0$ and $\frac{\partial F}{\partial b} = 0$ for a minimum, Equation (3) reduces to

$$a \Sigma x + b \Sigma y + n = o$$

or

$$a \left(\frac{\Sigma x}{n}\right) + b \left(\frac{\Sigma y}{n}\right) + 1 = 0,$$

which shows that Equation (1) is satisfied by the values

$$x = \left(\frac{\Sigma x}{n}\right) = \bar{x}, \quad y = \left(\frac{\Sigma y}{n}\right) = \bar{y}$$

In other words, the best representative line always passes through the centroid of the given points. The slope of the line is given by $\left(-\frac{a}{b}\right)$ where

$$b = -\frac{(\Sigma x)(a) + n}{\Sigma y}$$

and a is found by substituting this value of b in the following equation and solving:

$$a(a^2 - b^2)\Sigma xy + (a^2 - b^2)\Sigma y - 2ab\,\Sigma x - a^2b(\Sigma x^2 - \Sigma y^2) - bn = 0$$

The intercept of the line is given by $\left(-\frac{1}{b}\right)$.

The following objections have been raised against the "Scarborough method": first, there is no justification for minimizing the sum of the squares of the <u>perpendicular</u> deviations, and not the deviations in some other direction; and second, a more serious objection, the straight line obtained by this method is not invariant under change in the units of either variable.

The other methods for determining the functional relation between two variables, described by Wald (1940) and Bartlett (1949), respectively, use groupings of the variables. The basis of both methods is that, if the values can be separated into a few large distinct groups, the means of the variables within each group will be little affected by random variation, and the differences among the group means will be due to systematic variation. Roughly speaking, the method leads to consistent estimates if a gap in the distribution pattern of values of one of the variables is sufficiently distinct in the neighborhood of the group limits that a grouping based on observed values is equivalent to a grouping based on true values. Clearly, under these conditions, the differences between groups may therefore be attributed to some extraneous variates.

Wald's work involved the method of averages which assumes that the best representative line is that for which the algebraic sum of the residuals in each group is zero and consequently, the algebraic sum of all the residuals will be zero. It turns out that when x and y observations have weights in constant ratio, the method of averages is unbiased, and its statistical efficiency compares well with the method of least squares, at a considerable saving in labor.

Wald divides the data into two groups of equal size according to the magnitude of values of one of the variables, and takes the line joining the points of means of the two groups as an estimate of the functional relation. Thus the estimated slope (b) and intercept (a) are given by:

$$b = \frac{\bar{Y}_2 - \bar{Y}_1}{\bar{X}_2 - \bar{X}_1} \quad \text{and} \quad a = \bar{Y} - b\,\bar{X}$$

Bartlett's method improves on Wald in most instances by incorporating a suggestion made by Nair and Shrivastava and also Nair and Benerjee (1942), viz., the use of a slightly modified method which they called the "method of group averages" to give more efficient results than the method of averages. The method of group averages assumes the best representative line is that for which the algebraic sum of the residuals in each extreme group will be zero but the algebraic sum of all residuals will not be zero. For fitting a

straight line, Nair and Shrivastava, found that by plotting the points of mean values of x and y for the first one-third and the last one-third of the whole set of observations, arranged in order of magnitude of x, and by joining these mean points we get a better estimate of the straight line than any other two group means.

By model sampling, Nair and Banerjee, collected evidence that the method of group averages gives better estimates of a and b for the line $y = a + bx$, than the method of averages which Wald had put forward. However, they used the two extreme groups for both the location and slope of the functional line. Bartlett suggested using as one point the mean coordinates of the observations, $\bar{X}$ and $\bar{Y}$, just as in the least-squares method for the location of the fitted straight line and to use the two extreme groups to calculate the slope. Thus the slope (b) and the intercept (a) are given by

$$b = \frac{\bar{Y}_3 - \bar{Y}_1}{\bar{X}_3 - \bar{X}_1} \quad \text{and} \quad a = \bar{Y} - b\,\bar{X}$$

Generally speaking, the subdivision of data into two groups (Wald) or three groups (Bartlett) should be decided on the basis of which system will result in less bias for your particular set of observations.

Our Fortran Program is available upon request for computation of the slope of a straight line by the methods of Scarborough, Wald, and Bartlett. As numerical examples, we have estimated the functional relation for two cases by the three methods. The methods are applied first to a special case where the least-squares method is appropriate. This numerical example is the same one that Bartlett used to show the accuracy of his method.

The attached computer print-out sheet shows the resultant equations for the straight lines computed and gives the confidence intervals of the computed slopes, where appropriate. The 95 per cent confidence interval for slope b of the least

squares' line is given by the formula:

$$b \pm t_{\alpha/2,\ n-2}\ S_b$$

where

$$S_b = \frac{S_{y/x}}{S_x\sqrt{n-1}}.$$

For the "Scarborough method", the slopes of the two elementary regression lines limit the location of the true line. The slope of the line for the first case i.e. where the least squares is appropriate calculated by the "Scarborough method" is identical to that calculated by the least squares method because there are no deviations in the x variable of the test case. The smaller the x deviations relative to the y deviations, the nearer the line calculated by the "Scarborough method" to the elementary regression line of y on x, and in the converse case, the nearer the calculated line will be to the elementary regression line of x on y.

Wald's confidence interval method of assessing the accuracy of the slope of his line is solving the following formula for β:

$$(\bar{X}_2 - \bar{X}_1)^2\ (b - \beta)^2 = \frac{t^2_{\alpha/2,\ n-2}}{N-2}\ [S_y^2 - 2\beta S_{xy} + \beta^2 S_x^2]$$

This relation does not hold for a small number of observations.

Bartlett adapted Wald's confidence interval method to assess the accuracy of the slope of his line which results in an equation of the basic form:

$$(\bar{X}_3 - \bar{X}_1)^2\ (b - \beta)^2\ \tfrac{1}{2}\ k = t^2_{\alpha/2,\ n-2}\ [S_y^2 - 2\beta S_{xy} + \beta^2 S_x^2]$$

to which a modification is made to handle a small number of observations.

To summarize: when presented with a scatter of x - y observations, the often ignored least squares' specific

assumption that the independent variable, x, be either controlled or free from error - limits the applicability of the inferences drawn to predicting values of y for changing values of x.

The three methods described in this paper enable us to derive more information from the x and y observations even though both are masked by errors and the error variances of both x and y variables and the ratio of these error variances are all "unknowns". These three methods enable us to find a consistent estimate of the slope of the functional line when the errors in x and y are random variables subject to the following conditions: zero correlations with the true values of x and y; and the error in both observed variables are mutually uncorrelated.

Thus the primary purpose of this paper has been to place emphasis on the ideas and assumptions involved in estimating a functional relationship; the aim being to promote understanding of the available solutions to fitting a straight line to x and y data when both variables are subject to error.

CASE I

X	Y
1.00	15.87
2.00	17.78
3.00	19.52
4.00	21.35
5.00	23.13
6.00	24.77

LEAST SQUARES ELEMENTARY REGRESSION Y ON X

Y = 14.165 + (1.782) X

THE 95 PCT. CONFIDENCE INTERVAL FOR THE SLOPE : 1.782 +/- (.049)

LINE CALCULATED BY SCARBOROUGH METHOD

Y = 14.163 + (1.782) X

LEAST SQUARES ELEMENTARY REGRESSION X ON Y

Y = -7.943 + (.560) X

THE 95 PCT. CONFIDENCE INTERVAL FOR THE SLOPE : .560 +/- (.015)

LINE CALCULATED BY WALD METHOD

Y = 14.150 + (1.786) X

THE 95 PCT. CONFIDENCE INTERVAL FOR THE SLOPE : 1.812 +/- (.091)

LINE CALCULATED BY BARTLETT METHOD

Y = 14.168 + (1.781) X

THE 95 PCT. CONFIDENCE INTERVAL FOR THE SLOPE : 1.778 +/- (.074)

USE OF ORTHOGONAL POLYNOMIALS FOR CURVE FITTING

By S. S. Kuo
Professor of Applied Mathematics and
Director of Computation Center

University of New Hampshire

INTRODUCTION

In a recent book[1], the author has described a method for fitting orthogonal polynomials to a set of equally spaced data points. It was shown that the problems of ill-conditioning are eliminated. Ill-conditioning is usually associated with normal-equation approach when the degree of polynomials to be fitted is large.

The purpose of the present paper is twofold. In the first place, we shall show how the application of orthogonal polynomials can be extended to a set of unequally spaced data points. In this paper the orthogonal polynomials are represented by Chebyshev series. We shall then describe the flowchart and a tested FORTRAN program together with detailed illustrative examples. The sample input and output are included.

NORMAL-EQUATION PROCEDURE

The principle of least square can be applied to the problem to fit a given number of data by a polynomial in the form

$$Y = k_0 + k_1 x + k_2 x^2 + \ldots + k_m x^m \tag{1}$$

Basically a minimum value of S is required where

$$S = \sum_{i=1}^{n} (Y_i - y_i)^2 \tag{2}$$

and n is the total number of given data. By setting the following m+1 first derivatives to zero:

$$\frac{\partial S}{\partial k_0} = 0, \quad \frac{\partial S}{\partial k_1} = 0, \quad \ldots, \quad \frac{\partial S}{\partial k_m} = 0$$

we obtain a set of m+1 simultaneous linear equations, or normal equations:

$$[A][k] = [B] \tag{3}$$

where

$$[A] = \begin{bmatrix} n & \Sigma x_i & \Sigma x_i^2 & \cdots & \Sigma x_i^m \\ \Sigma x_i & \Sigma x_i^2 & \Sigma x_i^3 & \cdots & \Sigma x_i^{m+1} \\ \cdot & & & & \\ \cdots & & & & \\ \cdot & & & & \\ \Sigma x_i^m & \Sigma x_i^{m+1} & \Sigma x_i^{m+2} & & \Sigma x_i^{2m} \end{bmatrix}$$

$$[k] = \begin{bmatrix} k_0 \\ k_1 \\ \cdot \\ \cdot \\ \cdot \\ k_m \end{bmatrix}$$

$$[B] = \begin{bmatrix} \Sigma y_i \\ \Sigma x_i y_i \\ \cdot \\ \cdot \\ \cdot \\ \Sigma x_i^m y_i \end{bmatrix}$$

The values of k_0 through k_m can then be obtained by solving Eq. (3).

Unfortunately, this normal equation approach fails where Eq. (3) is ill-conditioned, often so when m is a large number.

USE OF ORTHOGONAL POLYNOMIALS

To overcome the difficulty of ill-conditioning mentioned above, a method designed for digital computers using orthogonal polynomials will be discussed.

Essentially, this method [2,3] fits the given data in the form:

$$Y_m(x) = C_0P_0(x) + C_1P_1(x) + \ldots + C_mP_m(x) \tag{4}$$

$P_m(x)$ is a polynomial of degree m having the following property:

$$\sum_{i=1}^{n} P_k(x_i)P_\ell(x_i) = 0 \qquad \text{for } k \neq \ell \tag{5}$$

The following recurrence relation is also useful:

$$P_{k+1}(x) = \lambda_k(x - \alpha_{k+1})P_k(x) - \beta_kP_{k-1}(x) \tag{6}$$

where

$$\beta_0 = 0$$

$$\beta_k = \frac{\lambda_k \sum_{i=1}^{n} \left[P_k(x_i)\right]^2}{\lambda_{k-1} \sum_{i=1}^{n} \left[P_{k-1}(x)\right]^2} ,$$

$$\alpha_{k+1} = \frac{\sum_{i=1}^{n} x_i \left[P_k(x_i)\right]^2}{\sum_{i=1}^{n} \left[P_k(x_i)\right]^2} ,$$

and λ_k are completely open to choice; thus no relationships exist between them. We chose $\lambda_k = 2$. If $P_0(x) = 0.5$ is specified, all $P_k(x)$ are completely defined for $k = 0, 1, \ldots, m$.

We now apply the least-square principle so that the expression

$$S = \sum_{i=1}^{n} \left[Y_i - \sum_{j=1}^{m+1} C_jT_j(x_i)\right]^2$$

is minimized. The resulting system of m+1 simultaneous equations can be written in the following matrix form:

$$\begin{bmatrix} \Sigma P_0^2(x_i) & 0 & \cdots & 0 \\ 0 & \Sigma P_1^2(x_i) & \cdots & 0 \\ \vdots & & & \\ 0 & 0 & \cdots & \Sigma P_m^2(x_i) \end{bmatrix} \begin{bmatrix} C_0 \\ C_1 \\ \\ C_m \end{bmatrix} = \begin{bmatrix} \Sigma Y_i P_0(x_i) \\ \Sigma Y_i P_1(x_i) \\ \vdots \\ \Sigma Y_i P_m(x_i) \end{bmatrix} \qquad (8)$$

where Σ implies $\sum_{i=1}^{n}$.

Obviously there is no need to solve the simultaneous equations and

$$C_j = \frac{\sum_{i=1}^{n} Y_i P_i(x_i)}{\sum_{i=1}^{n} P_j^2(x_i)} \quad , \qquad (j = 0, 1, \ldots, m) \qquad (9)$$

therefore, the problem of ill-conditioning is avoided.

CHEBYSHEV EXPANSION

Chebyshev polynomials of degree n in x are defined by

$$T_n(x) = \cos(n \cos^{-1} x) \qquad (10)$$

The first five Chebyshev polynomials are as follows:

$$T_0(x) = 1$$

$$T_1(x) = x$$

$$T_2(x) = 2x^2-1$$

$$T_3(x) = 4x^3-3x$$

$$T_4(x) = 8x^4-8x^2+1$$

Other Chebyshev polynomials can be readily obtained by using the

following recurrence relation:

$$T_{n+1}(x) - 2xT_n(x) + T_{n-1}(x) = 0 \tag{11}$$

It is possible to represent a given function F(x) by the following summation of Chebyshev series:

$$F(x) = \tfrac{1}{2}a_0 + a_1T_1(x) + a_2T_2(x) + \ldots + a_mT_m(x). \tag{12}$$

Clenshaw[4] has shown that

$$F(x) = \tfrac{1}{2}(b_0-b_2) \tag{13}$$

where b_0 and b_2 are obtained by using the following relations:

$$b_{m+2} = 0$$

$$b_{m+1} = 0$$

and

$$b_j = a_j + 2xb_{j+1} - b_{j+2} \tag{14}$$

$$(j = m, m-1, \ldots, 0)$$

REPRESENTATION OF $P_j(x)$ BY CHEBYSHEV EXPANSION

Equation (12) can now be used to represent the coefficients $P_j(x)$, $(j= 0, 1, \ldots, m)$ in Eq. (4), or,

$$P_j(x) = \tfrac{1}{2}p_0^{(j)} + p_1^{(j)}T_1(x) + p_2^{(j)}T_2(x) + \ldots + p_{j-1}^{(j)}T_{j-1}(x) + T_j(x) \tag{15}$$

where the coefficient of $T_j(x)$ term is arbitrarily set to one for the sake of normalization of $P_j(x)$, and the superscript in parenthesis denotes the degree of the polynomial to which the coefficient belongs.

Substituting Eq. (15) in Eq. (6), we have the following useful relations:

$$p_j^{(k+1)} = p_{j-1}^{(k)} + p_{j-1}^{(k)} - 2\beta_{k+1}p_j^{(k)} - \beta_k p_j^{(k-1)} \tag{16}$$

Finally the polynomials $Y_j(x)$ can be represented in the form of Eq. (12), or,

$$Y_j(x) = \tfrac{1}{2}A_0^{(j)} + A_1^{(j)}T_1(x) + A_2^{(j)}T_2(x) + \ldots + A_j^{(j)}T_j(x) \tag{17}$$

Comparing the coefficients of $T_j(x)$ in the following equation:

$$Y_k(x) = Y_{k-1}(x) + C_k P_k(x) \tag{18}$$

we have

$$A_j^{(k)} = A_j^{(k-1)} + C_k P_j^{(k)} \tag{19}$$

where C_k is expressed in Eq. (9).

It should be mentioned that when Eq. (9) is used in a straightforward manner, 4n storage locations are needed to store the following four items, each of n locations:

1. data x_i
2. data y_i
3. $P_k(x_i)$
4. $P_{k+1}(x_i)$

$$(i = 1, 2, \ldots, n)$$

The representation of $P_j(x)$ by Chebyshev expansion can generally save a substantial storage location in a digital computer.

CHANGE OF INTERVALS

In our discussion above, we have tactily assumed that the x_i values $(i = 1, 2, \ldots, n)$ all lie in the interval $(-1, 1)$. In practice, data are not necessarily so given, and a simple transformation will be needed.

If x denotes the data given in the interval (a,b), and $\bar{x}$ denotes the corresponding data in the interval (-1, 1), we have

$$\bar{x} = \frac{2x - (a + b)}{b - a} \quad . \tag{20}$$

A more involved task is transform all results from the interval (-1, 1) back to (a,b). This task requires a change of an entire function

$$Y = \sum_{j=1}^{m+1} A_j T_j(x) \tag{21}$$

In the first phase, the right hand side of Eq. (21) can be transformed to its power series equivalence without changing the interval:

$$Y = \sum_{j=1}^{m+1} D_j x^j \tag{22}$$

by using Eq. (11), the coefficients $p_k^{(j)}$ in the identity

$$T_j(x) = \sum_{k=0}^{j} p_k^{(j)} x^k \tag{23}$$

are found from the following equation:

$$p_k^{(j)} = 2p_{k-1}^{(j-1)} - p_k^{(j-2)} \tag{24}$$

and the coefficient for each x^k term is formed by successively adding the quantities $A_j p_k$. In other words, the coefficients D_j can be evaluated as the double sum:

$$Y = \sum_{j=1}^{m+1} A_j \sum_{k=0}^{j} p_k^{(j)} x^k \tag{25}$$

In the second and final phase, the power series expression in Eq. (25) is transformed back to the original given interval (a,b). This phase can be readily performed by setting

$$\bar{x} = \frac{x-\mu}{\sigma}$$

where

$$\mu = \frac{a+b}{2} \qquad \text{and} \qquad \sigma = \frac{b-a}{2}$$

In other words, the double sums in the following expression must be calculated:

$$\sum_{j=1}^{m+1} D_j \left(\frac{x-\mu}{\sigma}\right) = \sum_{j=1}^{m+1} \frac{D_j}{\sigma^j} \sum_{k=0}^{j} \binom{j}{k} (-1)^k x^{j-k} \mu^k \qquad (26)$$

The coefficients thus obtained are the final polynomial coefficients for the power series in the interval (a,b).

"BEST" FIT CRITERION

The polynomial of so called best fit may be determined by an examination of the quantity $\Delta = S/(n-j)$ when each degree, j, is being tried. In general, the Δ-value will increase first(as j-value increases) and then decreases. It is proposed that the program accommodates the Δ-value to increase and then decrease only once. A second increase of Δ-value will cause the program to stop.

FLOWCHART AND FORTRAN PROGRAM

The procedure presented in the previous sections is well suitable to electronic computation. A flowchart is shown in Fig. 1. The numbers shown in brakets are the corresponding statement numbers for a tested FORTRAN program which is listed in Fig. 2.

The input variables for the FORTRAN program are defined as follows:

IP = Degree of the polynomial Y_m desired. If no particular degree is desired and a best fit is required, IP = 99.

M = Number of original data points.

X(I) = Value of the original data x_i.

Y(I) = Value of the original data Y_i.

JJ = Program stopper, when JJ $\geq$ 2.

EXAMPLE

It is required to fit a two-term (IP = 1) polynomial to the following four data points:

x	0	1	2	3
y	7	12	11	16

In the same run, it is also required to fit a polynomial six-degree to the following 20 data points:

x	1	3	4	5	6	7	8	9	10	11	11.5	12
y	2	7	8	10	11	11	10	9	8	3	3	14

13.5	14	15	16	16.5	17.5	18	20
6	3	20	-4	-4	0	3	6

Applying the FORTRAN program to this specific example, the data would be read in the format as shown in Fig. 3.

The output for the Chebyshev program gives the coeffcients A_j for the polynomial Y in terms of the power series:

$$Y_m = \sum_{j=1}^{m} A_j x^j$$

In addition, the given points farthest above and farthest below the least squares curve are also printed out. They are listed in Fig. 4.

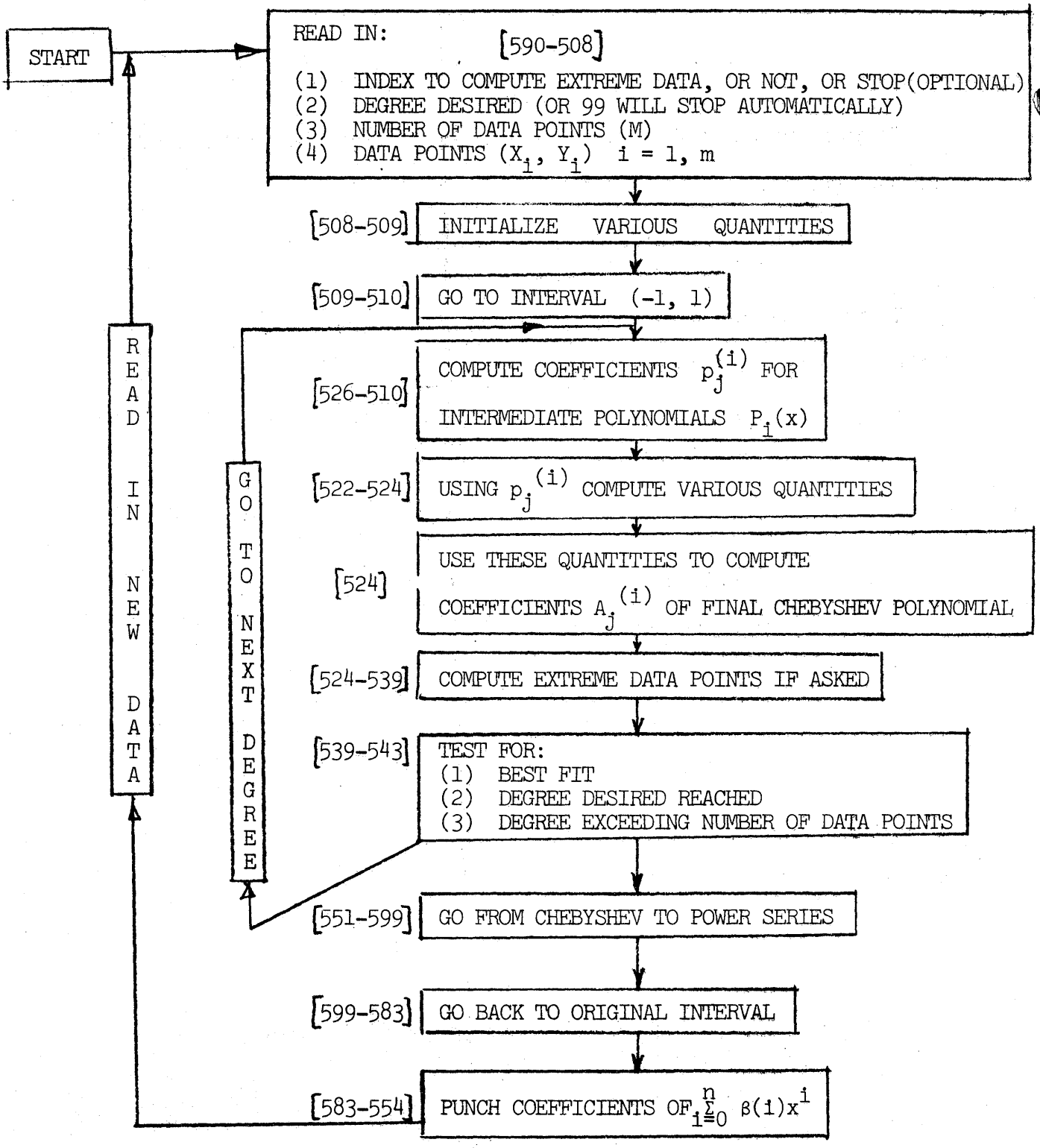
BIBLIOGRAPHY

1. Kuo, S. S., NUMERICAL METHODS AND COMPUTERS, Addison-Wesley Publishing Co., Reading, Massachusetts, 1965. Chapter 11.

2. Lanczos, C., APPLIED ANALYSIS, Prentice-Hall Co., Englewood Cliff, New Jersey.

3. Clenshaw, C. W., "Curve Fitting With a Digital Computer," British Journal of Computer, January, 1960, pp. 170-173.

4. Clenshaw, C. W., "A Note on the Summation of Chebyshev Series," Mathematical Tables and Computations, Vol. 9, 1955, p. 118.



FLOWCHART FOR LEAST SQUARES CURVE FITTING USING

ORTHOGONAL POLYNOMIALS + CHEBYSHEV SERIES
(Program Statement in Brackets)

THE POINTS FARTHEST ABOVE AND FARTHEST
BELOW THE LEAST SQUARES CURVE ARE

DEGREE	ABOVE	BELOW
1	(0.10000001E+01, 0.12000000E+02)	(0.20000000E+01, 0.11000000E+02)

POLYNOMIAL COEFFICIENTS - DEGREE 1

A(0) = 0.76000010E+01
A(1) = 0.25999998E+01

THE POINTS FARTHEST ABOVE AND FARTHEST
BELOW THE LEAST SQUARES CURVE ARE

DEGREE	ABOVE	BELOW
1	(0.15000000E+02, 0.20000000E+02)	(0.16000000E+02,-0.40000000E+01)
2	(0.15000000E+02, 0.20000000E+02)	(0.16000000E+02,-0.40000000E+01)
3	(0.15000000E+02, 0.20000000E+02)	(0.16000000E+02,-0.40000000E+01)
4	(0.15000000E+02, 0.20000000E+02)	(0.16000000E+02,-0.40000000E+01)
5	(0.15000000E+02, 0.20000000E+02)	(0.16000000E+02,-0.40000000E+01)
6	(0.15000000E+02, 0.20000000E+02)	(0.16000000E+02,-0.40000000E+01)

POLYNOMIAL COEFFICIENTS - DEGREE 6

A(0) = 0.54671820E+01
A(1) = -0.73307400E+01
A(2) = 0.48346740E+01
A(3) = -0.10214046E+01
A(4) = 0.97473200E-01
A(5) = -0.43494959E-02
A(6) = 0.73559148E-04

Curve Fitting by Finite Sum of Exponentials

Richard Mishelof

Supervisor, Bio-Computing Section

Computing Center

State University of New York - Downstate Medical Center

Brooklyn, New York

Nallur Prasad

Computer Applications Inc.

Introduction

Data which usually arises from tracer experiments, stress fatigue in metals, diffusion of gases, etc. and which is more often then not unequally spaced in time (time considered the independent variable), is approximated by a function of the form:

$$g(t) = \sum_{j=1}^{N} C_i e^{-\alpha_i t} + C_\infty \qquad (1)$$

We will restrict our analysis for α: positive, real and that all the α's are well separated; Ci's are real and C_∞ is considered to be zero. The order N, the number of exponential components, in the summation, is usually dictated by the mathematical model. If no information is available on what the mathematical model should be, then the model can be evolved from the data.[7] This is the usual procedure in compartmental analysis.

Our approach is to get initial values for α's by a semi-graphical procedure known as "peeling-off". These initial values are our starting point for an iterative scheme which we hope will refine our initial values.

Discussion of Methods (Fitting Techniques)

1. Algebraic Method:

   Prony's method can be applied.[1,2] The values of 2N+1 points which are evenly spaced in time t are input to a set of simultaneous linear equations of rank N+1

which yields the Coo and the N+1 coefficients to a polynomial of order N. The roots of polynomial are α_i which are used as input to develop a further set of simultaneous linear equations which yield the Ci values.

It has been the authors' experience that the answers one gets are erroneous (I'm being very mild to say the least). The values for α_i have been negative and complex. One may interpolate to get evenly-spaced data, but the results are not worth the effort put into it. This method is not recommended.

II. Derivative Peel-Off Procedure

This method is described by Perl.(2) In essence it is very similar to the method I use, and may be better for some applications to get the initial starting points for iterative procedure. We consider Coo=o; this can be done by having a previous knowledge of Coo and subtracting it from g(tj) j=1,...,M points.

A portion of our curve can be represented by $q_1(t) = C_1 e^{-\alpha_1 t}$ (2)

if we take the derivative, we have $-\dot{q}_1(t) = C_1 \alpha_1 e^{-\alpha_1 t}$ (3)

(3)

We can then determine the slope $q_1(t)$ vs $\dot{q}_1(t)$ which is calculated numerically under the t considered. This gives us a value for α_1 and the ordinate - intercept gives $C_1\alpha_1$ thus C_1. We then subtract q_1 from q for all tj j=1,....,M and repeat the procedure. This procedure will give good estimates for Ci, α i i=!,..,N, if our data warrants a derivative approximation calculation.

III. The Author's Approach

A. "Peeling-off" type of procedure (semi-graphical)

1. Redefinition of problem:

We are given points $t_1, t_2, \ldots, t_m$ not necessarily evenly spaced with corresponding $q(ti) = q_1, q_2, \ldots, q_m$.

We wish to approximate our data to a finite sum of exponentials. Our approach is as follows:

2. The natural log of the data points qi is taken and the result is placed into an array di. Thus,

$$di = \ln(qi) \qquad i = 1,\ldots,M \qquad (4)$$

3. We start with the Mth value of di and work backwards and approximate a least square straight line

$$y = atj + b \qquad j = M,\ M-1,\ldots,1 \qquad (5)$$

where a and b are the solutions to the set of normalized equations.

$$\begin{aligned} a\sum_{J=K_1}^{M-K}\frac{t_j^2}{d_j} + b\sum_{J=K_1}^{M-K}\frac{t_j}{d_j} &= \sum_{J=K_1}^{M-K} t_j \\ a\sum_{J=K_1}^{M-K}\frac{t_j}{d_j} + b\sum_{J=K_1}^{M-K}\frac{1}{d_j} &= M-(K+K_1-1) \end{aligned} \qquad (6)$$

which means we are working on the k_1 to k interval of points of the M points of our set of observations. We take the measure of our error to be expressed in the form:

$$E = \sum_{J=K_1}^{M-K}\frac{1}{d_j}(y-d_j)^2 \qquad (7)$$

which is a pseudo-least square residual. It should be noted that there are many other error criteria which we could try to minimize for example:

$$E' = \sum_{J=K_1}^{M-K}\frac{1}{d_j^2}(y-d_j)^2 \qquad (8)$$

which had a tendency to produce a Jacobian of our iteration procedure which was of the order of 10^{-4} and hence a nearly ill-conditioned system of equations.

$$E'' = \sum_{J=K_1}^{M-K} \ln\left[\frac{q_j}{\sum_{i=1}^{n} C_i e^{-\alpha_i t_j}}\right] \tag{9}$$

see reference Worsley for a comprehensive discussion of equation (9) as well as a polynomial approximation to the data.

4. This marching backward procedure terminates when we have: (1) and error, as measured by equation (7), greater than a certain maximum error that we arbitrarily chose, in our case, .05; (2) run out of data points. It should be noted that this arbitrary maximum allowable error is a function of M and dj j=1,...,M and the true α_i, Ci values.

5. We next form a new set of data dj j=1,...,M which is

$$d_j = d_j - a t_j - b \tag{10}$$

and our initial estimates for Ci α i are

$$\begin{aligned} \alpha\ i &= -a \\ Ci &= e^b \\ \text{increment} \quad & i+1 \rightarrow i \end{aligned} \tag{11}$$

6. The above procedure is run continuously until we:

   (1) run out of data points; (2) have gotten the number of exponentials we expect: N

B. We now go into our iteration procedure with our initial guesses α_i, Ci and will treat all the data points. In the previous section we treated data points where we had a "slow α " (minimum α of unknown set of α's). As stated above we chose our error E to be of the form

$$E = \sum_{J=1}^{M} \frac{1}{q_j}\left[\sum_{i=1}^{N} C_i e^{-\alpha_i t_j} - q_j\right]^2 \tag{12}$$

we wish to minimize $E(\alpha_1, \alpha_2, \ldots, \alpha_N, C_1, C_2, \ldots C_N; q_j, t_j) = E$

The critical point $p=(\alpha_1,\alpha_2,\cdots,\alpha_N,C_1,C_2\cdots,C_N)$ where the maximum or minimum of E lies is either in the boundary of $D(\alpha, C)$ or on the boundary, where

$$D(\alpha, C) = \begin{cases} -\infty < C_i < \infty \\ 0 \leq \alpha_i < \infty \end{cases} \tag{13}$$

Appealing to the calculus,[5] we know that the 2N partial derivative of E with respect to α_i and C_i $i=1,\ldots,N$ will give the local critical point if equated to zero.

$$\frac{\partial E}{\partial \alpha_1} = \frac{\partial E}{\partial \alpha_2} = \cdots\cdots = \frac{\partial E}{\partial \alpha_N} = \frac{\partial E}{\partial C_1} = \frac{\partial E}{\partial C_2} = \cdots = \frac{\partial E}{\partial C_N} = 0 \tag{14}$$

or the 2N system of simultaneous non-linear algebraic equations.

$$\begin{aligned} 0 = B_\ell = \frac{\partial E}{\partial \alpha_\ell} &= \sum_{j=1}^{M} \frac{t_j}{q_j} e^{-\alpha_\ell t_j} \left[\sum_{i=1}^{N} C_i e^{-\alpha_i t_j} - q_j \right] \\ 0 = B'_\ell = \frac{\partial E}{\partial C_\ell} &= \sum_{j=1}^{M} \frac{e^{-\alpha_\ell t_j}}{q_j} \left[\sum_{i=1}^{N} C_i e^{-\alpha_i t_j} - q_j \right] \end{aligned} \qquad \ell = 1,\ldots,N \tag{15}$$

There are several iteration techniques for solving a system as the above.[3,6] We chose the Newton-Rapshon technique. Our experience with convergence patterns for various sets of data showed the initial estimates need not be close to the solution for convergence. The continuity of the solution vector, implicit in the Newton-Rapshon method, is a criterion for convergence. For instance, if the first two or three data points do not show sufficient drop, we found it impossible to make the iteration converge until the data point which did not drop enough was deleted.

Conclusion

The algebraic method for finding the solution to equation (1) gives erratic results, the polynomial approximation for the set data points does not give a satisfactory result.

Graphical techniques as described above give fairly good initial estimates and an improvement by a pseudo-least square procedure has proven successful in most cases tried by the authors.

The data should be extended out in time such that each α_i is represented by the original curve or extrapolated curves. α_i that are very close together may be lumped together and still give a satisfactory error.

Gardner[7] has investigated the numerical inversion of the Laplace transform, which appears to be a method for finding the true number of exponentials. This method has proven relatively successful with evenly-spaced generated data. There are a number of problems to be overcome with experimental data.

The methods described in this paper will not solve all problems. Some other approaches might prove more successful. These other solutions can be found in the references given if only to look-up their references (indirect addressing).

We have had a certain amount of success with our problem and approaches and hope that follow up work in field will do much to improve what may be termed by some as an art:

The art of knowing how to approximate your data to finite sum of exponentials.

REFERENCES

1. Hildebrand, Introduction to Numerical Analysis, McGraw-Hill, page 378-382.

2. Perl, International Journal of Applied Radiation and Isotopes, 1960, Vol. 8 pages 211-222.

3. Traub, Iterative Methods for the Solution of Equations, Prentice-Hall, pages 248-257.

4. B. Worsley and L. Lax, Selection of Numerical Techniques for Analyzing Experimental Data of the Decay Type with Special Reference to the Use of Tracers in Biological Systems, Biochimica et Brophysica Acta, Vol. 59, No. 1, pages 1-24.

5. Buck, Advanced Calculus, McGraw-Hill pages 285-298.

6. Wolfe, The Secant Method for Simultaneous Non-Linear Equations, Comm. ACM Vol.2, pages 12-13.

7. Gardner etal, Method for Analysis of Multicomponent Exponential Decay Curves, Journal Chemistry and Physics Vol. 31, No. 4, page 978.

APPENDIX

DERIVATION OF NORMALIZED EQUATION FOR ERROR

We chose our error E as defined by equation (7)

$$E = \sum_{J=K_1}^{M-K} \frac{1}{d_j}(at_j + b - d_j)^2 \qquad (1)$$

taking the partial derivative of E with respect to a and b, and equating to zero we get

$$\frac{\partial E}{\partial a} = \sum_{J=K_1}^{M-K} \frac{2}{d_j}(at_j + b - d_j)t_j = 0$$

$$\frac{\partial E}{\partial b} = \sum_{J=K_1}^{M-K} \frac{2}{d_j}(at_j + b - d_j) = 0 \qquad (2)$$

if we carry through with the summation, we get the normalized form of equation (6)

$$a\sum_{J=K_1}^{M-K} \frac{t_j^2}{d_j} + b\sum_{J=K_1}^{M-K} \frac{t_j}{d_j} = \sum_{J=K_1}^{M-K} t_j$$

$$a\sum_{J=K_1}^{M-K} \frac{t_j}{d_j} + b\sum_{J=K_1}^{M-K} \frac{1}{d_j} = M - (K+K_1-1) \qquad (3)$$

Newton-Rapshon Iteration Procedure and Application to our System of Equations

We are given a system of equations

$$f_i(x_1^*, x_2^*, \ldots, x^*n) = 0 \qquad i=1,\ldots,n \qquad (1)$$

where $x_1^* \ldots x^*_n$ is the solution to our system.

We let

$x_1^\circ, x_2^\circ, \ldots\ldots, x_n^\circ$ be the initial estimate for our solution, such that

$$
\begin{aligned}
x^*_1 &= x^\circ_1 + \Delta x_1 \\
x^*_2 &= x^\circ_2 + \Delta x_2 \\
&\vdots \\
x^*_n &= x^\circ_n + \Delta x_n
\end{aligned}
\tag{2}
$$

We approximate the system by a Taylor expansion and we get

$$
o = f_1(x^*, \ldots, x^*_n) = f_1(x, x, \ldots, x_n) + \sum_{i=1}^{n} \frac{\partial f_1}{\partial x_i}\Bigg|_{x^\circ_1, x^\circ_2, \ldots, x^\circ_n} (\Delta X) + \text{terms } (\Delta X)^P
$$

$$
o = f_n(x^*, \ldots, x^*_n) = f_n(x, x, \ldots, x_n) + \sum_{i=1}^{n} \frac{\partial f_n}{\partial x_i}\Bigg|_{x^\circ_1, x^\circ_2, \ldots, x^\circ_n} (\Delta X_i) + \text{terms } (\Delta X)^P
$$

We assume that all $(\Delta X)^P$ where $P \geq 2$ is approximately zero

If we put the system of equations (3) in to matrix form, we get

$$
-[F] = [J][X] \tag{4}
$$

where

$$
[F] = \begin{bmatrix} f_1(x_1 \ldots xn) \\ f_2(x_1 \ldots xn) \\ \vdots \\ fn(x_1 \ldots xn) \end{bmatrix} \qquad \Delta X = \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \vdots \\ \Delta xn \end{bmatrix} \tag{5}
$$

and the Jacobian matrix

$$
[J] = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_m} \\ \vdots & & \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_m} \end{bmatrix} \tag{6}
$$

We solve system (4) for ΔX and set

$$
\begin{aligned}
x'_1 &= x^\circ_1 + \Delta x_1 \\
x'_2 &= x^\circ + \Delta x_2 \\
&\vdots \\
x' &= x^\circ_n + \Delta xn
\end{aligned}
\tag{7}
$$

If the determinant of the Jacobian Matrix remains non-zero and

$$\lim_{K' \to \infty} [F] = [0] \tag{8}$$

where K' is iteration count, then system is said to converge

Our system of equations are defined by equations (15) that is

$$0 = B_\ell = \frac{\partial E}{\partial \alpha_\ell} = \sum_{J=1}^{M} \frac{t_j}{q_j} e^{-\alpha_\ell t_j} \left[\sum_{i=1}^{N} C_i e^{-\alpha_i t_j} - q_j \right]$$

$$0 = B'_\ell = \frac{\partial E}{\partial C_\ell} = \sum_{J=1}^{M} \frac{e^{-\alpha_\ell t_j}}{q_j} \left[\sum_{i=1}^{N} C_i e^{-\alpha_i t_j} - q_j \right] \tag{9}$$

where $\ell = 1, 2, \ldots, N$

thus vector (F) is partitioned into two vectors

$$F = \begin{bmatrix} B_\ell \\ B'_\ell \end{bmatrix} \tag{10}$$

Let us now develope the Jacobian Matrix where the terms are:

$$\frac{\partial B_\ell}{\partial \alpha_K}, \frac{\partial B_\ell}{\partial C_K}, \frac{\partial B'_\ell}{\partial \alpha_K}, \frac{\partial B'_\ell}{\partial C_K} \qquad \text{for } \ell, K = 1, 2, \ldots, N$$

For $\frac{\partial B_\ell}{\partial \alpha_K}$ where $\ell \neq K$

$$\frac{\partial B}{\partial \alpha_K} = \sum_{J=1}^{M} \frac{(-1) t_j^2}{q_j} e^{-\alpha_\ell t_j} C_K e^{-\alpha_K t_j} \tag{11}$$

For $\frac{\partial B_\ell}{\partial \alpha_K}$ where $\ell = K$

$$\frac{\partial B_K}{\partial \alpha_K} = \sum_{J=1}^{M} \frac{(-1) t_j^2}{q_j} e^{-\alpha_K t_j} \left[\left(\sum_{i=1}^{N} C_i e^{-\alpha_i t_j} - q_j \right) + C_K e^{-\alpha_K t_j} \right] \tag{12}$$

If we borrow Kronecker's delta, then we can generalize the above results.

$$\delta_{ij} = \begin{cases} 0.0 & i \neq j \\ 1.0 & i = j \end{cases} \tag{13}$$

$$\frac{\partial B_\ell}{\partial \alpha_K} = \sum_{J=1}^{M} \frac{(-1) t_j^2}{q_j} e^{-\alpha_\ell t_j} \left[\delta_{\ell,K} \left(\sum_{i=1}^{N} C_i e^{-\alpha_i t_j} - q_j \right) + C_K e^{-\alpha_K t_j} \right] \tag{14}$$

$$\frac{\partial B_\ell}{\partial C_K} = \sum_{J=1}^{M} \frac{t_j}{q_j} e^{-\alpha_\ell t_j} e^{-\alpha_K t_j}$$

$$\ell, K = 1, 2, \cdots, N$$

Once again appealing to Kronecker delta we generalize $\frac{\partial B'_\ell}{\partial \alpha_K}$

$$\frac{\partial B'_\ell}{\partial \alpha_K} = \sum_{J=1}^{M} \frac{(-1) e^{-\alpha_\ell t_j}}{q_j} \left[\delta_{\ell,K} \left(\sum_{i=1}^{N} C_i e^{-\alpha_i t_j} - q_j \right) + C_K e^{-\alpha_K t_j} \right] \tag{15}$$

and

$$\frac{\partial B'_\ell}{\partial C_K} = \sum_{J=1}^{M} \frac{e^{-\alpha_\ell t_j} e^{-\alpha_K t_j}}{q_j} \tag{16}$$

$$\ell, K = 1, 2, \cdots, N$$

Our Jacobian Matrix is partitioned into four parts

$$[J] = \begin{array}{c} \\ B_\ell \\ \\ B'_\ell \\ \\ \end{array} \left[\begin{array}{c|c} \frac{\partial B_\ell}{\partial \alpha_K} & \frac{\partial B_\ell}{\partial C_K} \\ \hline \frac{\partial B'_\ell}{\partial \alpha_K} & \frac{\partial B'_\ell}{\partial C_K} \end{array}\right] \begin{array}{c} \\ \alpha_K \end{array} \quad \begin{array}{c} \\ C_K \end{array} \tag{17}$$

PROGRAM OUTLINE

A program for the method discussed in this paper was written in FORTRAN II-D. There was no attempt to economize in core space, or in execution time. There are several variables that could be made equivalent (i.e. EXPØN, ALPHA), (CØEFF, C)), and a DØ loop which could be incorporated into a following set of DØ loops.

The program uses three subprograms one SUBRØUTINE and two FUNCTIØN programs.

SUBRØUTINE SØNYA (N, A, B, X, DET)

Is a program for solving a system of simultaneous linear equation AX=B by pivotal condensation method. The technique is described by Faddeeva, Computational Methods in Linear Algebra, a Dover publication, and was applied to a system of ill-conditioned simultaneous linear equation in Wilkinson, Rounding Errors in Algebraic Processes, Prentice-Hall, page 118-120, with a high degree of accuracy.

The program is designed to solve a system of 20 x 20, but can be expanded or shortened by changing the proper arrays in the DIMENSIØN statements. The terms are:
N is the rank of the coefficient matrix A, A is the coefficient matrix, B constant vector, X solution vector, DET is the determinant of matrix A.

FUNCTIØN DELTA (I, J)

Is the Kronecker delta (see paper)

FUNCTIØN SUMEX (NEXP, TAU, C, ALPHA)

Is a program to evaluate the sum of exponentials $\sum_{i=1}^{N} C_i e^{-\alpha_i \tau}$ where NEXP is the number of exponential components.

TAU is dummy variable, C, ALPHA are respective coefficient and exponential multiplier. It should be noted that:

$e^{-19} = 5.60279\ 64375 \times 10^{-9}$ and is considered to be zero, if expanded accuracy is needed then the value 19 should be enlarged. This test prevents exponential underflow.

The entire system of programs was designed to handle a maximum of 5 exponential components, but this restriction can be altered by expanding the proper arrays of all the system programs.

TYGO - A load-and-go version of S.P.S.

R.C. Read and P.J. Jutsum

Programmers who use S.P.S. a lot on a papertape machine, must frequently get impatient at the amount of time that is consumed in punching and handling tape, using the standard two-pass processor, before a program can be compiled. TYGO (TYPE-and-GO symbolic programming system) has been devised for the benefit of these programmers. It enables an S.P.S. program to be typed in at the console and assembled directly in storage without the punching and handling of any tape being necessary apart from the loading of the processor tape. However input of the source statements can be from tape if desired, and an object tape can be produced (under operator control) at any time after compiling, - before and/or after running or debugging the program.

Thus in some ways TYGO is to S.P.S. as GOTRAN is to FORTRAN; but whereas GOTRAN is a fairly restricted subset of FORTRAN, TYGO is effectively as powerful as S.P.S., except for some restriction on the possible length of the program, since the processor occupies about 9000 locations at the top of storage during compilation. The only other significant restriction in TYGO is that multiplication is not admissible in address arithmetic. All the standard S.P.S. subroutines can be called, and also the floating part input/output subroutines (Users Group Library No. 1.6.023).

Since in the source program symbols may be used before they are defined, a more complicated symbol table is needed than in regular S.P.S. Undefined symbols are stored together with the addresses at which the equivalent (when it becomes known) is to be put. During compilation a listing of the program can be produced, in which undefined operands are left blank; a listing of the symbol table can be produced when compilation is finished.

The symbol table grows downwards from the beginning of the processor (location 11000 or thereabouts). Impending overlap with the program is signalled, and a new pseudoinstruction DEL enables unwanted portions of the symbol table to be removed to make room for more program. After compilation the processor area can be used for subroutines and/or data areas. Programs of up to about 8000 locations will compile easily and compilation is faster than with the usual S.P.S. processor (SP-008).

TYGO is written for the basic 20K tape 1620 with no special features.

SPAS

G.G. Billin

SPAS is a modification of 1620/1710 SPS[1] designed for LOAD-and-GO operation. It is best used for debugging programs or running programs that will not be used frequently. Because of its usage, several additions and deletions have been made to the basic SPS language. Some features of SPAS are:

* No cards except source statements are punched. This means a large savings in cards while the program is being debugged.

* The object program is ready to be run at the end of PASS II with no intermediate loading of an object deck.

* The processor may be made one pass by turning on a sense switch. The information on each card is stored in upper memory during PASS I and is recalled as needed during PASS II. Thus, no time is lost while the reader reads a card, and the cards are only read once.

* The processing time, exclusive of the one pass feature, has been reduced.

The machine requirements for SPAS are indirect addressing and a 40K or 60K card 1620 system. Of these requirements, only the indirect addressing feature may be modified.

SPAS was written because of time and card limitations at the computer center where the author worked. It was decided first of all to eliminate all punched card output; next, the speed of the processor was increased, new operations were added, and other miscellaneous modifications were made.

Since the author used a card 1620 rather than a 1710 and because macro-instructions were seldom used, the following mnemonic operation codes were deleted from the list of 1620/1710 operations:

All macro-instructions:

FA	FSQR	FEXT	FSRS
FS	FCOS	FEX	FSLS
FM	FSIN	FLOG	TFLS
FD	FATN	FLN	BTFS
DIV			

All 1710 control operations:

SLRN	SAO	MK	UMK	BO
SA	SACO	SAOS	SLTA	BOLD
SLCB	SLTC	SLAD	SLME	SLAR

1 1620/1710 SPS, version 2, modification 15.

All paper tape operations:

RNPT WNPT DNPT RAPT WAPT

In addition, the following SPS instructions were modified:

DNB lengths may be greater than 50 to a maximum of 1000.

TRA assembles the instruction:

```
49 02468 00000
```

where 02468 is the address of the PASS II read instruction.

TCD loads the following record into location 00000:

```
NOP            00000  41 00000 00000
B    xxxxx     00012  49 xxxxx
```

where xxxxx is the address specified by the TCD instruction. Next, control is transferred to location 00000.

SEND halts the processor. When START is pushed, processing continues.

The following operations were added:

BRC	Branch if Read Check (06)
BNRC	Branch No Read Check
BWC	Branch if Write Check (07)
BNWC	Branch No Write Check
BEC	Branch if Check in MBR-E (16)
BNEC	Branch No Check in MBR-E
BOC	Branch if Check in MBR-0 (17)
BNOC	Branch No Check in MBR-0

The following operations generate no output in the object program.

MESS	Return carriage and type message starting in column 16.
MES1	Same as MESS except on PASS I only.
MES2	Same as MESS except on PASS II only.
TYPE	Type message starting in column 16 (no carriage return).
TYP1	Same as TYPE except on PASS I only.
TYP2	Same as TYPE except on PASS II only.
KM	Perform carriage control (34 00000 0010X) operation with digit in column 16 replacing X instruction.
LOD1	Simulates LOAD during PASS I only.
LOD2	Simulates LOAD during PASS II only.

These instructions were designed to be used as follows:

MESS, MES1, MES2, TYPE, TYP1, TYP2 - To type headings at the beginning of each pass, to type special messages to the operator, or to provide an indication of how far a program has progressed.
SEND - To stop so that switches may be changed.
LOD1, LOD2 - To allow loading of changes to the processor, the program being assembled, or to transfer control directly to the program.

An example of the use of the new instructions is as follows:

SOURCE PROGRAM

```
MES2SPAS
MES2DATED
TYP2 3/
TYP218
TYP2/65
MES2TURN SWITCH 1 ON FOR LISTING, PLEASE
SEND
```

During PASS II the typewriter will type:

```
SPAS
DATED 3/18/65
TURN SWITCH 1 ON FOR LISTING, PLEASE
```

and the computer will stop. When START is pushed, the computer will continue processing.

MODIFICATIONS TO THE SPS PROCESSOR

The following modifications have been made to the SPS processor to make it compatible to one pass and LOAD-and-GO operation:

The initial address of the address counter is set at 20000 rather than at 00402. This was done on the assumption that the first module of storage would be used for the processor and symbol table. To change this address, make the first card of the source program a DORG card. Be sure not to accidently DORG over the processor. SPAS does not check for this--it is the responsibility of the programmer.

All references to page and line number of statements have been eliminated from the processor.

When listing a source program on the typewriter during PASS II, the usual format included a carriage return and a tab between the source statement line and the assembled instruction. With SPAS this carriage return is not performed unless the source statement goes past column 56 on the card. This may be changed

back to the original way by putting a 0 (zero) at location 1787. To restore the feature later, put a 1 (one) at the same location.

A subroutine to count the number of carriage returns and to skip 5 lines at the top and bottom of each page has been included. The proper initial setting of the typewriter carriage should leave a margin of 6 lines at the top of the page.

When assembling in one pass, the approximate number of source statements possible is about 800, assuming that locations 40000 to 59999 are used for storing records.

If one pass mode and typewriter input have been selected, there will be no punched output during PASS I. For two pass mode, the punched output will be the same as for SPS with the exception of the deletion of page and line numbers as noted above.

The actual running of a program in SPAS is approximately the same as for SPS. One exception is the use of switch 3. If it is on, one pass mode has been selected. If it is off, processing is two pass. After this interrogation at the beginning of PASS I, the setting of the switch is disregarded. The only difference is one additional error--number 15. This error message is typed out like any other one to indicate that the records put in upper memory have overflowed the area assigned to them. No more processing can be done after this message has been typed--the only thing to do is to return to PASS I (by pushing START) or to change the size of the area where records are stored.

In PASS II several changes have been made. Switch 3 is not interrogated any more--all output is automatically put in memory. If one pass mode has been selected the only action necessary is to push START.

Another modification can be found at the end of PASS II. A listing of the symbol table can be made as usual if switch 4 is on, however, if switch 3 is on after the halt after the symbol table listing, control will go to location 00000 where the record:

```
H             00000  48 00000 00000
B   xxxxx     00012  49 xxxxx 00000
```

was placed at the end of PASS II. xxxxx is the address specified by the DEND.

DESCRIPTION OF SPAS PROCESSOR

SPAS is a modification of the SPS processor, therefore many of the changes are designed to cover up old instructions; these changes will not be discussed. The new features of the SPAS processor can be broken up into the following separate routines: initialization for PASS I, one pass storage of re-

cords, return carriage subroutine, numerical blanks routine, high and low positions used, output routine, special operations routine, and the end of PASS II routine.

Initialization for PASS I

The first instruction executed resets the carriage return subroutine for the correct margin for the new page. Next switch 3 is checked. If it is on, one pass operation has been selected and the following record is put at 02468 in place of

02468 31 00796 59999
02480 12 02479 00005

These two instructions will cause the records stored in upper memory during PASS I to be transmitted into the input area during PASS II. If switch 3 is off, operations will be two pass and the original instruction to read a card and a NOP are put at 02468 and 02480.

* This address 59999 can be changed by the user if desired--it is usually as shown.

One Pass Storage of Records

After the symbol table has been cleared at the beginning of PASS I, the first statement is read from the card reader or the typewriter depending on the setting of switch 1 and is saved in the INPUT2 area. Next, control returns to the original SPS program and the op code, label, constant and symbol lengths are checked to see that they are ok. When the SPS processor is done, control passes back to SPAS at location 02116. If two pass mode has been selected, switch 1 is interrogated and if it is off the caved source statement is punched. Next, control goes to the read instruction. If one pass mode has been selected the card is scanned for the third comma in it or a record mark. The comma would indicate the end of information used by the processor--all after it is comment. An exception is a DAC or DSA statement which is not scanned for commas since any number of commas may may be present.

When the address of the third comma or record mark has been determined, the program checks to see if the statement will fit in upper memory. Two pieces of data are stored for each record, the record itself and a field that tells where its high order digit is. The records are stored from the low address up and are variable length, while the location fields are stored from the high address down and are a fixed, five digit address. If the length of the record about to be entered is greater than the difference between the last used low address and the last used high address, the ER 15 message is typed and the processor HALTS. If there is enough room for the record, it is transmitted into the lower address and that address is stored at the higher address. The program then returns to the read instructions. If the statement just read was a DEND statement, the DEND statement is stored as usual but the processor goes to the end of PASS I instructions.

Return Carriage Subroutine

A return carriage subroutine, RCTYPE, has been included so that long listings on the 1620 typewriter can be done conveniently.

Entry to the subroutine is by branching and transmitting the return address to the address of the subroutine minus one. In this way entry can be done with a BTM or by a TFM and branch if the return carriage occurs while the processor is already in a subroutine.

Numerical Blanks Routine

SPS assembles numeric blanks by punching them alphamerically on a card. Since no cards are punched in SPAS, this function is simulated by transmitting numeric blanks one by one to the address given or assigned by the processor minus the number of blanks defined, up to the specified address. The routine to do this starts at location 06736. The length of the output (at location 00704) is subtracted from the address assigned (at location 01122). Then blanks are transmitted until the assigned address is reached.

High and Low Positions Used

Before each object instruction or constant is stored in memory, a routine checks to see if its lowest and highest locations used are the lowest and highest used so far. As the object program is stored, the routine keeps track of these addresses and at the end of PASS II they are typed out. This is to let the programmer know how long his program is and to inform him whether he has gone below 20000 or into some other area.

Output Routine

The output routine works almost the same as the SPS loader, except that it uses indirect addressing to save space. It is located at location 06018 (PCHCRD). First the digit that will over with a record mark is saved. Then the record is transmitted after finding out where in the INPUT2 area it starts, and the saved digit is replaced.

Special Operations

The special operations routine replaces the macro-instructions routines. The processing for special operations starts at location 09672. Each operation in SPS or SPAS is referred to by an eleven digit field of which 8 digits represent the alphanumeric coding of the mnemonic operation, 2 digits represent the op code or indicator, and the last digit represents the type. Special operations are type -7. The digit preceding the type digit defines the particular procedure while the second digit represents whether it occurs during PASS I or PASS II in the case of singular operations. The identifying digits are as follows:

0 MESS
1 MES1,MES2
2 TYPE
3 TYP1,TYP2
4 LOD1,LOD2
5 KM

When a special operation is found, control goes to 09672. There the type is determined and control passes to the proper routine by means of a table look up using a DSA. A MESS operation causes the carriage to return and the operation:

WATY INPUT+20

to be executed. In a like manner, MES1 or MES2 first checks to see if it is the proper pass and then either executes the MESS routine or bypasses it. A similar procedure is followed for the TYPx routines except that the carriage return is not done. LOD1 and LOD2 cause a simulated load during the corresponding pass and when KM is encountered, the digit at INPUT+20 (the units digit of the alphameric code) is transmitted to the Q_{11} position of a control instruction which is then performed. If the digit transmitted is other than a 1, 2, or 8, some combination of the functions will be performed.

End of PASS II Routine

At the end of PASS II, control is transferred to location 12060. There, the lowest and highest addresses used are transmitted to an available area (INPUT2), a record mark is put after them, and they are written out. Next, the record that originally went on the last card of the SPS loader is transmitted to location 00000. Control then passes to the SPS routines that write:

End of PASS II

and the symbol table is listed if switch 4 is on. At the end of this the computer HALTS. When START is pushed, control passes to either the start of PASS I or to location 00000 depending on the setting of switch 3.

As mentioned earlier, SPAS uses indirect addressing and does not include macro-operations. Both of these problems may be gotten around, the first by reprogramming, the second by a special technique.

To reprogram SPAS for a machine not equipped with indirect addressing, it would be necessary to change several instructions in SPAS. SPS uses no indirect addressing. To make matters easier space has been provided between locations 05440 and 06016 for any additional instructions. By changing SPAS for a non-indirect addressing machine, approximately 500 locations would be used. Of course, new operations could be processed by routines in this area.

To use macro-instructions with SPAS the best thing to do would be to assemble them at some high address or in the space

used by the SPAS symbol table. Then to enter a macro-instruction it would be necessary to simulate the instructions generated by the SPS processor when macros are used with it.

This is at most a sketchy report on SPAS. The program is to be submitted to the Users' Group in the near future so that it will be easily available. For the moment, copies of the decks and documentation are available on a trial basis to groups who are willing to report to the author on the effectiveness of the program. The author's address is:

Mr. Geoffrey G. Billin
Computer Center
Clarkson College
Potsdam, New York 13676

ABSTRACT FOR FN II WORKSHOP

Understanding Fortran II
(SPS Subroutines called from Fortran II)

by

L. Hoffman
Guggenheim Laboratories
Princeton University

To increase the unfullness of Fortran II for large, long problems, one must realize which aspects of FN II take the most time and storage. A few examples are iterative procedures, subscripting, and special output formats. Often, iterative procedures are done as subprograms, such as solution of differential equations. These subprograms can be written in SPS to give an object subprogram which will usually occupy one-half of the equivalent FN II subprogram and usually will take about one-half the execution time of the FN II version.

In order to present the technique of SPS subprograms, the standard linkage generated by FN II is described and a variation of the linkage for many arguments is presented. A description of FN II non-relocatable subroutines is given. A sample SPS printer-plot subprogram is given as an example of linkage, communication, and relocatability of an SPS written subroutine.

NON-LINEAR ESTIMATION
BY
MODIFIED GAUSS-NEWTON METHOD

BY
JERRY KEMP
RALPH CHIPMAN
CHARLES BRYANT

Presented to the IBM 1620 USERS Group at Miami Beach, Florida on
May 11, 1965

TABLE OF CONTENTS

APPENDIX

PART

A	General Flow Diagram
B	Fortran Source Statements
C	Partial Printout
D	Complete Printout

NON-LINEAR ESTIMATION BY MODIFIED GAUSS-NEWTON METHOD

I. INTRODUCTION

This paper presents the modified Gauss-Newton method for the fitting of non-linear regression functions by least squares as described by Hartley (1). In Section II, a mathematical discussion is presented and in Section III the computer work is described.

Frequently, experimenters are faced with determining a functional relation between a response (dependent variable) and a number of inputs (independent variables) with the help of empirical data. Usually the mathematical form is assumed to be known and the coefficients of the inputs (unknown parameters) must be estimated.

II. MATHEMATICAL DISCUSSION

Before discussing non-linear regression, it is appropriate to make a few comments on linear regression.

A. Linear Regression

If the parameters in the mathematical model are linearly related, the least squares estimates are obtained by direct solution of simultaneous equations.

1. Simple Linear Regression

An example is as follows:

$$y_i = a + bx_i + e_i \tag{1}$$

Where y_i is the measured response to a measured input, x_i

a and b are parameters to be estimated

e_i is the error or lack of fit

i is the observation number

Least squares estimates are defined such that $\sum e_i^2$ is a minimum. From here on, let $Q = \sum e_i^2$. To find the least squares estimates, set $\frac{\partial Q}{\partial a} = 0$ and $\frac{\partial Q}{\partial b} = 0$. This yields the following equations which are solved simultaneously for a and b.

(2)
$$\begin{aligned} Na + b\sum x_i &= \sum y_i \\ a\sum x_i + b\sum x_i^2 &= \sum x_i y_i \end{aligned}$$

Where N is the number of observations and $\sum$'s are summations from 1 to N. (Note linearity in a and b).

2. Multiple Linear Regression

An example is as follows

(3)
$$y_i = a + bx_i + cx_i^2 + e_i$$

Where a, b, and c are parameters to be estimated

While this model is non-linear in x_i, it is still linear in the parameters (a, b, and c). Setting $\frac{\partial Q}{\partial a} = 0$, $\frac{\partial Q}{\partial b} = 0$, and $\frac{\partial Q}{\partial c} = 0$, yields the following set of simultaneous equations which are linear in a, b, and c.

(4)
$$\begin{aligned} Na + b\sum x_i + c\sum x_i^2 &= \sum y_i \\ a\sum x_i^2 + b\sum x_i^3 + c\sum x_i^4 &= \sum x_i y_i \\ a\sum x_i^2 + b\sum x_i^3 + c\sum x_i^4 &= \sum x_i^2 y_i \end{aligned}$$

B. Non-Linear Regression

When the parameters are not linearly related, we are faced with non-linear estimation for which we have no direct solution. Hartley's article uses the following model as an example.

(5)
$$y = f(x; L, B, K) = L + Be^{Kx}$$

$$y_i = L + B\exp(Kx_i) + e_i$$

Where y_i is the measured response to the input (x_i) for the i th observation

L, B, and K are parameters to be estimated

and e_i is the error or lack of fit for the i th observation

To illustrate the difficulty with this function, set $\frac{\partial Q}{\partial L} = 0$ and examine this result

(6)
$$\frac{\partial Q}{\partial L} = 2\sum(y-f)(-f_1) = 0$$

Where y is the observed value, f is the function, and f_1 is the $\frac{\partial f}{\partial L}$

Substitute $L + B \exp(Kx_i)$ for f, and 1 for f_1 in (6). This yields

$$(7) \qquad 2\sum\left[y_i - (L + B \exp (Kx_i)) \right]\left[-1 \right] = 0$$

which may be rewritten as

$$\sum y_i - NL - B \sum \exp (Kx_i) = 0$$

In $\sum \exp (Kx_i)$, K is "locked"inside of the summation and even the usual techniques for solution of simultaneous non-linear equations are not applicable.

Since no direct solution is possible, we are forced to revert to the approximation process which is described next.

C. Non-Linear Iteration Technique

This non-linear estimation technique uses the Newton-Raphson method (Reference 2, Page 463) for defining a correction to apply to the parameter estimates from the previous iteration. The Newton-Raphson Method use a Taylor Series approximation. Values for these corrections are solved for in the Gauss Newton equations. Hartley's modification guarantees that once the iterations begin to converge the process will not diverge at a later time. For the process to converge, starting values for the parameters must be "close" to the "true" values due to the Taylor Series approximation.

D. Sample Problem and Solution

1. The Problem

From the data presented in the following table determine least squares estimates of L, B, and K in equation (5).

OBSERVATION NUMBER	x	y
1	-5	127
2	-3	151
3	-1	379
4	1	421
5	3	460
6	5	426

Trial values

Lo = 580
Bo = -180
Ko = -.160

2. The Solution

Values of corrections (D1, D2, D3) to these trial values are obtained by solution of the following simultaneous equations.

$$(8)\quad D_1 \sum(f_1)_o^2 + D_2 \sum(f_1f_2)_o + D_3 \sum(f_1f_3)_o = \sum(y-f)_o (f_1)_o$$

$$D_1 \sum(f_1f_2)_o + D_2 \sum(f_2)_o^2 + D_3 \sum(f_2f_3)_o = \sum(y-f)_o (f_2)_o$$

$$D_1 \sum(f_1f_3)_o + D_2 \sum (f_2f_3)_o + D_3 \sum(f_3)_o^2 = \sum(y-f)_o (f_3)_o$$

where

$$f_1 = \frac{\partial f}{\partial L} = 1$$

$$f_2 = \frac{\partial f}{\partial B} = e^{Kx}$$

$$f_3 = \frac{\partial f}{\partial K} = Bxe^{Kx}$$

y-f = observed y minus predicted y

$(\quad)_o$ refers to evaluation using L = Lo

B = Bo and K = Ko

$\sum$ - Summation over all observations

Solve for v (min) in the following equation (parabolic fit)

$$(9)\quad v\ (\min) = 1/2 + 1/4\ (\ Q(0) - Q\ (1)\ )\ /\ (Q\ (1) - 2Q\ (1/2) + Q\ (0)\ )$$

where

$$Q = \sum(y-f)^2 = \sum e_i^2$$

Q (0) = Q with L = Lo, B = Bo and K = Ko

Q (1/2) = Q with L = Lo + 1/2 D_1

B = Bo + 1/2 D_2

K = Ko + 1/2 D_3

Q (1) = Q with L = Lo + D_1

B = Bo + D_2

K = Ko + D_3

Then the inital values for the next iteration are

(10)
$$L_1 = Lo + vD_1$$

$$B_1 = Bo + vD_2$$

$$K_1 = Ko + vD_3$$

If the process is converging, the value of Q using L_1, B_1, and K_1 will be less than the value of Q using Lo, Bo, and Ko. If not, try v = 1/2 v(min.) If the process still diverges, try new trial values of L, B, and K. If the process converges, the iterations are continued until ΔQ is less than some small arbitrary value (ϵ)

(11) Where $\Delta Q = \frac{Q_k - Q_{k+1}}{Q_{k+1}}$ (k denotes the iteration number)

If ϵ is too small, then D_1, D_2 and D_3 approach zero which might cause divergence due to computer accuracy.

The computer print-out of the solution is shown in Parts C and D of the Appendix. In this solution, ϵ was set at .0001. The standard error (stder) equals the square root of Q/(M-N).

The computer details are presented in the next section.

E. Other Examples

The following two equations are examples of our work with this technique.

Interior Ballistics (LeDuc equation)

$$v = \frac{au(w+c)^d}{b+u}$$

where v = Muzzle velocity

u = barrel length

w = propellant charge weight

and a, b, c, and d are unknown parameters to be estimated

Solid State Burning Rate Equation

$$u = \frac{P^n (a+b \ln P)}{c+d \ln P}$$

where u = burning rate

P = Pressure

n, a, b, c, and d are chemical and physical parameters to be estimated.

III. DEFINITION OF COMPUTER PROGRAM AND OPERATIONAL INSTRUCTIONS

A. General

This is a one-pass program, written in PDQ Fortran with fixed format subroutines. The program was compiled on an IBM 1620 (20K) computer with automatic divide. The input is on cards and the output is on the typewriter. The general flow diagram, Fortran source statements, and sample printout are contained in the Appendix.

B. Program Limitations

1. The general form of the dimension statement is as follows:

Dimension A((N+2),(N+1)),PD(N),C(N),D(N),CNST(N),Q(5),Y(M),T1(M),---TK(M)

The only limitation to this dimension statement is given by the equation

$$N^2 + 7N + K(M+1) < 473$$

where N = number of parameters
M = number of observations
K = number of independent variables and
N < M

C. Source Statement Changes: Once the problem is defined; make the following changes to the source statements when necessary:

1. Dimension statement: The dimension statement must be changed to reflect

(N) the number of parameters
(M) the number of observations and
(K) the number of independent variables

as dictated by the problem at hand.

2. Read statement: The changes made for reading the dependent and independent variables is a function of the number of independent variables. These changes should be made so that Y(M) is the dependent variable and T1(M)---Tk(M) are labeled as the independent variables.

For example: Read 5, T1(I), ------, TK(I), Y(I)

3. Procedure 1: The changes made here are used in solving for Q and the partial derivatives in procedure 2. The last statement in procedure 1 should be DIFF = Y(K)-U, where U is equal to Y(K) calculated. Make the changes so the number of arithmetic operations are at a minimum.

4. Procedure 2: This procedure evaluates the partial derivatives, PD(1) through PD(N) where N is the number of parameters.

D. Input Data Preparation

1. Punch source statement changes

2. Punch data as follows:

a. Header Card (punched in Floating Point)

Identification	Columns
Number of observations	1 - 10
Number of parameters	11-20
ϵ (small arbitrary value) which determines termination of iteration	21-30

b. Observations (punched with decimal)

Card 1 - M (where M is the number of observations) punch the independent and dependent variables in columns 1-10, 11-20, 21-30, etc., depending on the number of independent variables. Make sure the dependent variables is punched so that it will be labeled as Y(I), and the independent variables are labeled as T1(I), TK(I), etc. The order is defined in the Read statement.

3. Parameters

The parameters (starting values) are punched once per card. They can be in F format, Col. 1-10, with decimal, or in E format ($\pm$ .xxxxxxxxE$\pm$xx) in cols. 1-14.

E. Computer Operator Instruction

1. Insert new source statements in program.

2. Compile program using PDQ Fortran Processor.

3. To execute object program:

a. Clear machine

b. Load program and PDQ Fortran Fixed format subroutines

c. When computer types Load data

d. Set Switch 1 for printout desired

1 off to print Q for each iteration
1 on to get full printout

e. Read in data in the following order: Header card, observations, and parameters

f. When computer comes to manual light, push start key if error check is desired, otherwise you're finished.

F. Example Problem

$$\frac{\partial U}{\partial C_1} = PD(1) = 1$$

$$\frac{\partial U}{\partial C_2} = PD(2) = e^{C_3 T}$$

$$\frac{\partial U}{\partial C_3} = PD(3) = C_2 T e^{C_3 T}$$

1 dependent and 1 independent variable

DIMENSION STATEMENT:

```
DIMENSION A(5,4), PD(3), C(3), D(3), CNST(3), Q(5), Y(6), T(6)
```

READ STATEMENT:

```
Read 5, T(I), Y(I)
```

PROCEDURE 1:

```
X1=C(3)*T(K)
X2=EXP(X1)
X3=C(1)+C(2)*X2
U = X3
```

PROCEDURE 2:

```
PD(1) = 1.
PD(2) = X2
PD(3) = C(2)*T(K)*X2
```

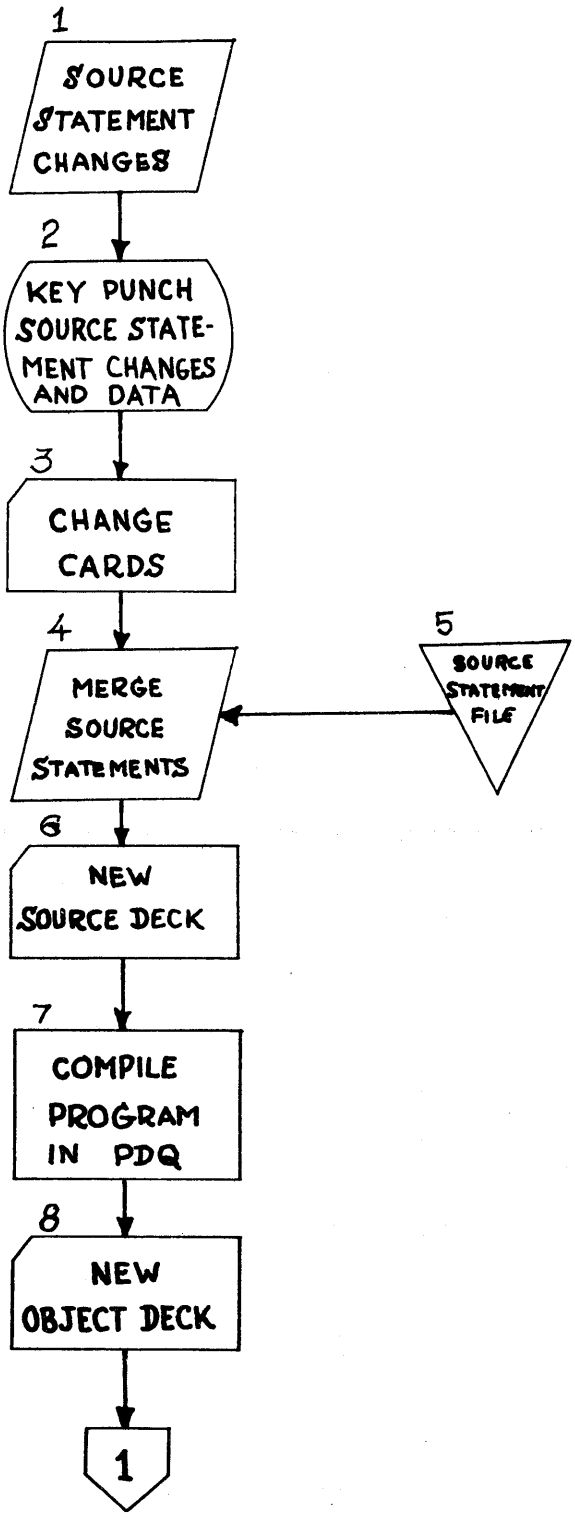
IV. REFERENCES

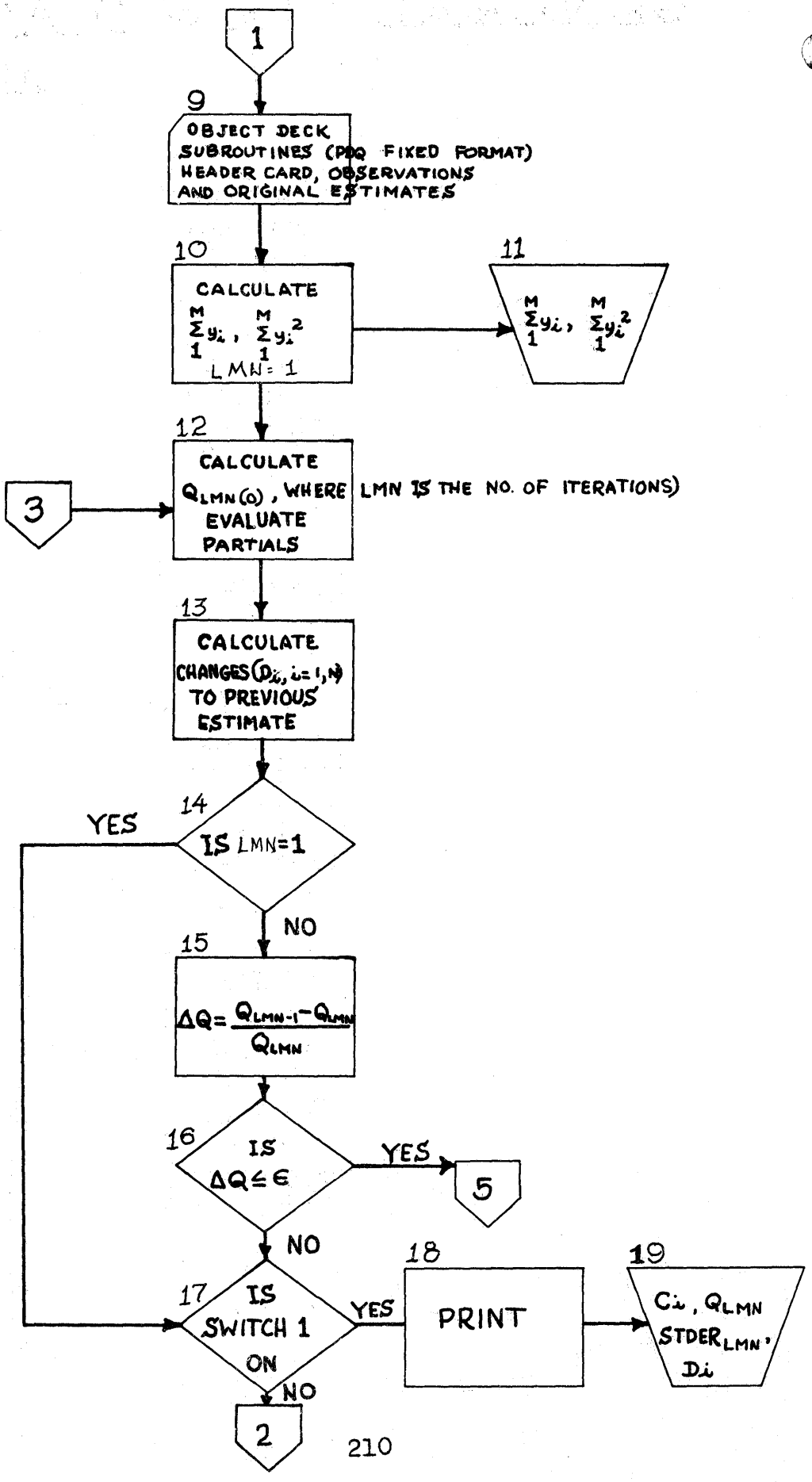
1. Hartley, H. O. "Technometrics", Vol 3 No. 2, page 269, May 1961

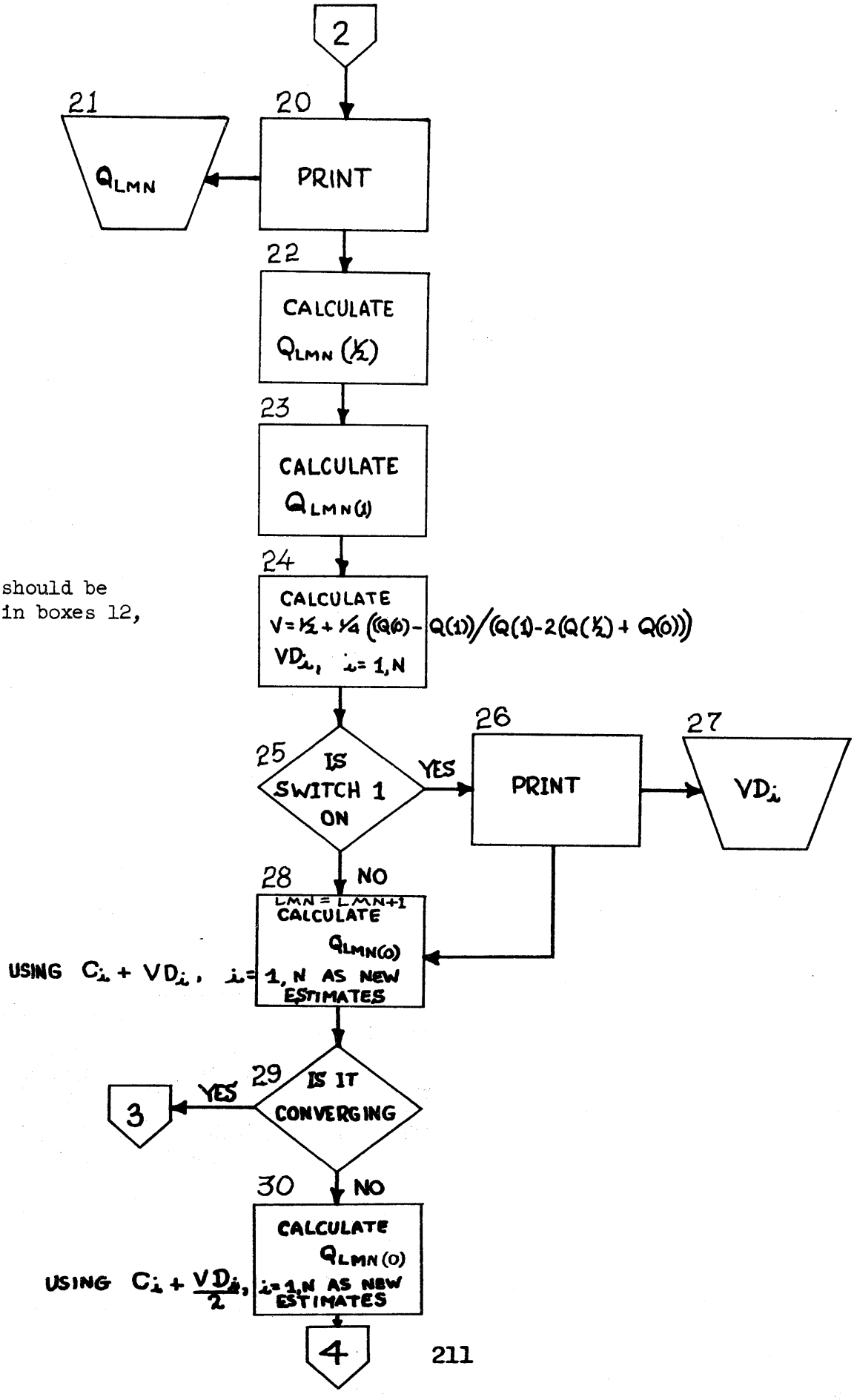
2. Scarborough, J. B. "Numerical Mathematical Analysis", 3rd Ed., The Johns Hopkins Press, Baltimore, Md., 1955

GENERAL FLOW DIAGRAM

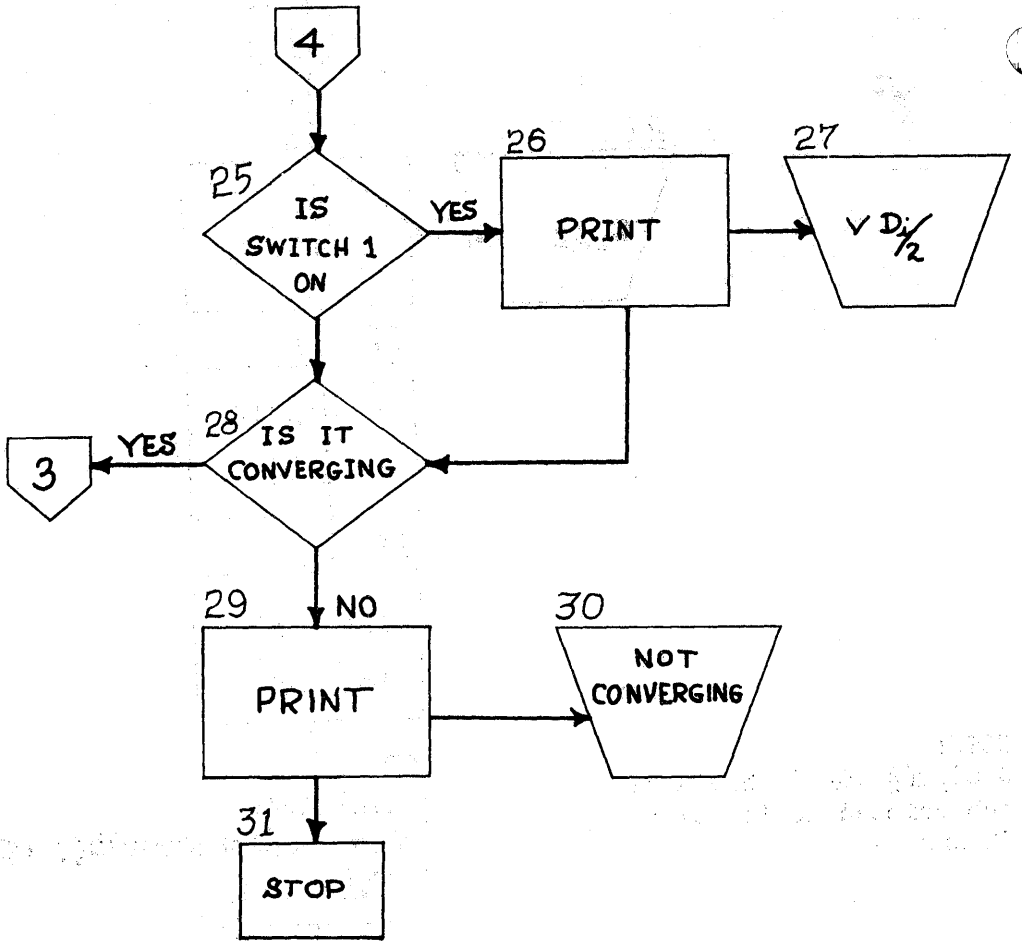
APPENDIX-PART A

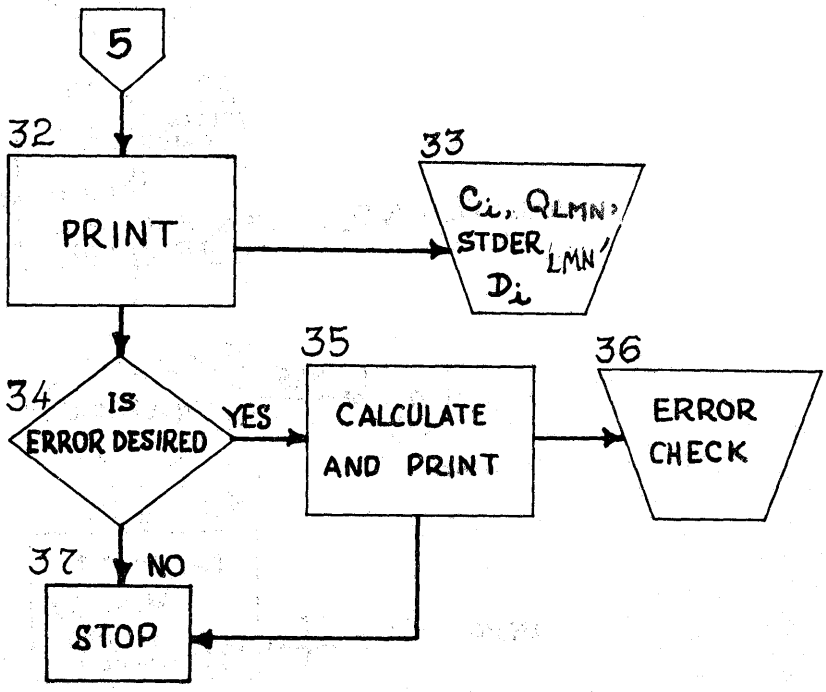



1
9
OBJECT DECK
SUBROUTINES (PDQ FIXED FORMAT)
HEADER CARD, OBSERVATIONS
AND ORIGINAL ESTIMATES
10
CALCULATE
$\sum_1^M y_i$, $\sum_1^M y_i^2$
LMN = 1
11
$\sum_1^M y_i$, $\sum_1^M y_i^2$
12
3
CALCULATE
$Q_{LMN}(Q)$, WHERE LMN IS THE NO. OF ITERATIONS)
EVALUATE
PARTIALS
13
CALCULATE
CHANGES (D_i, i = 1,N)
TO PREVIOUS
ESTIMATE
14
YES
IS LMN=1
NO
15
$\Delta Q = \frac{Q_{LMN-1} - Q_{LMN}}{Q_{LMN}}$
16
IS
$\Delta Q \leq \epsilon$
YES
5
NO
17
IS
SWITCH 1
ON
YES
18
PRINT
19
C_i, Q_{LMN}
$STDER_{LMN}$,
D_i
NO
2



NOTE:
$Q(0), Q(\frac{1}{2}), Q(1)$ should be subscripted as in boxes 12, 22 and 23.


4
25
IS SWITCH 1 ON
YES
26
PRINT
27
V Di/2
3
YES
28
IS IT CONVERGING
29
NO
PRINT
30
NOT CONVERGING
31
STOP


5
32
PRINT
33
Ci, QLMN, STDER LMN, Di
34
IS ERROR DESIRED
YES
35
CALCULATE AND PRINT
36
ERROR CHECK
37
NO
STOP

BUCKNELL UNIVERSITY

LEWISBURG, PENNSYLVANIA

Freas-Rooke Computing Center

THE AUTOLOG-DIGIKLOK SYSTEM

by

Edward F. Staiano
Director, Freas-Rooke Computing Center
Daniel E. Atkins
Senior Assistant
May 1, 1965

ABSTRACT

An online digital clock, although available for most large computers, has not been readily available for the IBM 1620. It is the belief of the authors that a 1620 with extended memory and disk storage is large enough to justify the development and use of an online digital clock in an automatic logging and program timing system. This paper describes the AUTOLOG-DIGIKLOK System currently in use at the Freas-Rooke Computing Center, Bucknell University.

The paper is presented in two parts. The first deals with the design and construction of the digital clock, the hardware of the system; while the second describes the program modification to Monitor I, the software of the system.

Part one reviews the possibilities considered by FRCC to obtain a digital clock and the device eventually built at Bucknell. The two major components of the clock, the timing section and the gating-interface section are described at the block diagram level, together with a cursory description of their construction.

Part two of the paper describes the software modifications and additions to the Monitor I system. The main software modification incorporates the clock into a system which generates a complete chronological record of computer use. The revised software as well as the actual operating systems are treated in detail.

The possibilities for using the AUTOLOG software without an online clock are discussed.

I. INTRODUCTION

A digital clock is a device that counts time in the form of numerical digits rather than as positions of hands on a dial. When these digits are represented electronically, the clock can be connected to an electronic digital computer as an online real time input device. A reading of the actual time may then be obtained by the programmer by instructing the machine to interrogate the I/O channel to which the clock is connected. This data may then be used to accomplish such tasks as automatic logging of jobs run on the computer, generation of random numbers, computing the elapsed time for program segments or entire programs, etc.

This paper is concerned with the development of a digital clock and the associated software for an IBM 1620 computer.

The body of the paper is divided into two sections. The first section is primarily concerned with the design and construction of the digital clock. It begins with a survey of commercially available clocks and continues with a block diagram description of the design of a working device. The discussion centers around the logic design of the timing circuits and the gating interface necessary for connection to the 1620 computer.

The second section of the paper describes the Autolog software developed for use with the digital clock. The software is discussed from the standpoint of the user and as a program modification to the Monitor I System. No attempt has been made to discuss secondary program additions such as a random number generator subroutine although they are available. These are standard SPS subroutines written for addition to the Fortran II-D library.

Since many users will not have a digital clock available for use with the Autolog program a section has been included that discussed the use of the Autolog Program without a digital clock. A short discussion of compatability with FORGO-D is included for those persons using the FORGO-D Monitor System.

II. THE DIGITAL CLOCK

Introduction

The hardware portion of the Autolog-Digiklok System is an online digital clock. This device, nicknamed the Digiklok, will maintain real time, and upon instruction from the IBM 1620 will transfer a clock reading through the 1620 I/O channel to core memory. This section briefly describes action taken by the Freas-Rooke Computing Center (FRCC) in obtaining a clock and the general operation of the Digiklok which was eventually designed and built at Bucknell University.

Obtaining the Clock

Several options for obtaining a digital clock were considered. IBM was first contacted and responded with an estimated cost of $15,000 for a complete custom built unit. A formal logic design was then prepared at the FRCC and submitted to several companies with requests for quotations on a complete unit, logic hardware necessary for construction of the clock at Bucknell, or some combination of these two options. The responses to these requests are presented in Table 1.

Although formal quotations were requested only for a solid state device, the possibility of constructing the clock with electro-mechanical switches was considered. It was decided, however, that the noise and maintenance demands of the electro-mechanical system coupled with the small price difference between switches and 100 Kcps solid-state logic justified the construction of a completely solid-state device. For the purposes of logging and program timing, a resolution of 0.1 second and the average accuracy of the power line frequency were considered sufficient, and thus the AC line was selected as a time base.

After considering the possibilities outlined in Table 1, the EECo, Q-Series modules were selected with the idea of completing detailed design and construction of the clock at Bucknell. The Q - Series hardware consists of a universal circuit card (Figure 1) and three different welded, encapsuled digital modules (Figures 2, 3, 4) which may be soldered to cards to implement standard digital circuits such as gates, squaring amplifiers, multivibrators, etc.

COMPANY	ITEMS INCLUDED IN QUOTATIONS	PRICE	ESTIMATED TOTAL COST	COMMENTS
IBM	Complete Unit		$15,000	
Tech Serv, Inc.	Logic cards, card files, 1 spare each type card. Without power supply.	$1700.00	$ 2,200	Not NOR/NAND Logic 4 card files required
Engineered Electronics, Co. (EECO)	5 options for the necessary logic and card files. Without spares. With power supply.			
	T-Series	$2991.70	$3200	100 kc logic
	CT-Series	$2534.55	$2750	Fully constructed
	G-Series	$1817.75	$2050	cards
	Q-Series Clamped	$1363.30	$1600	100 kc or 25 kc
	Q-Series Unclamped	$1295.10	$1400	cards built from sub modules
NAVCOR	No definite price offered.			
C.E. Snow	No definite price offered.			
PARABAM	Constructed basic clock without interface, gating, or parity.	$2720.00	$3700	Estimated cost for constructing gating at Bucknell: $1000.

TABLE 1 - Results of Requests for Quotations

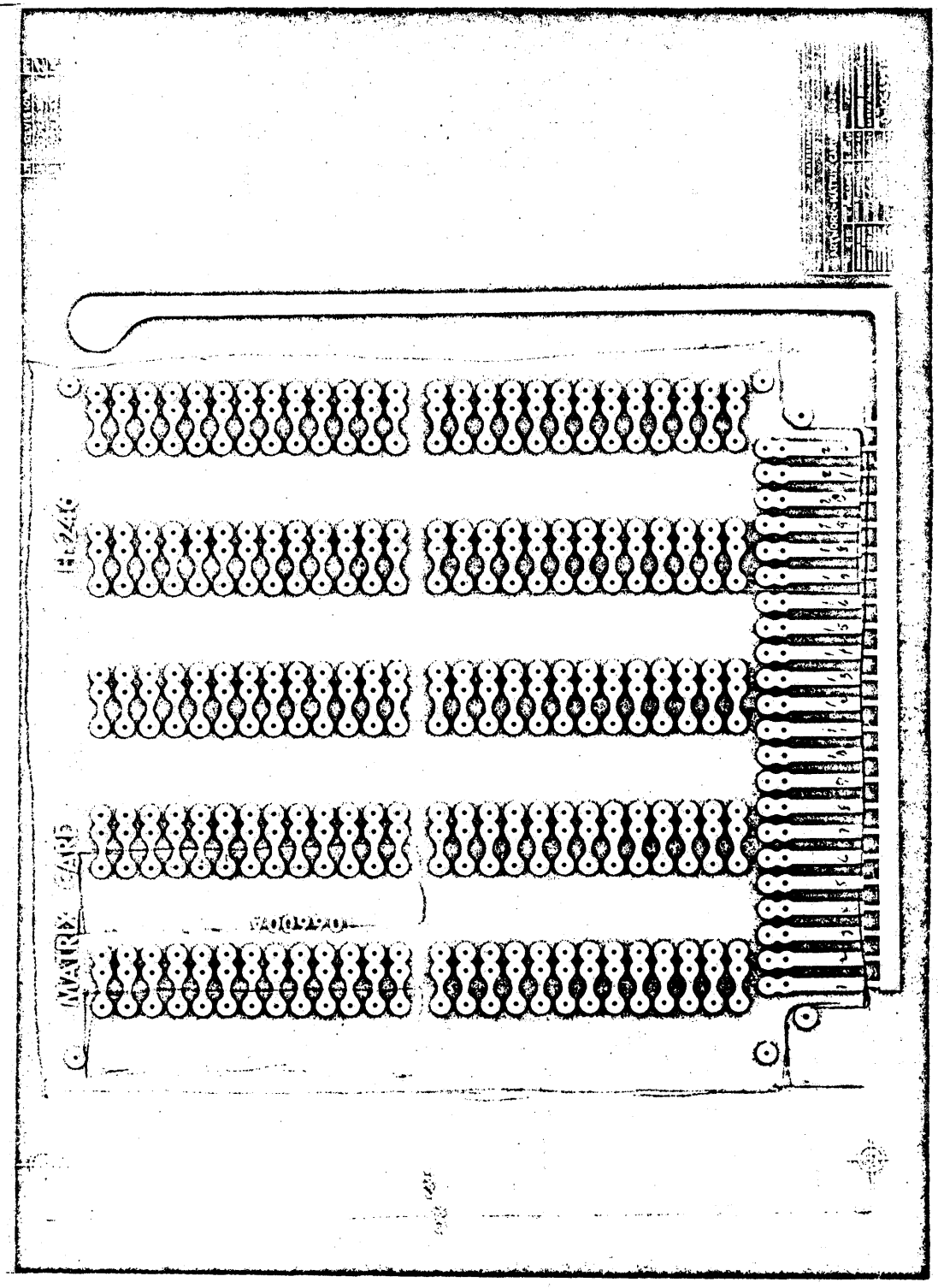
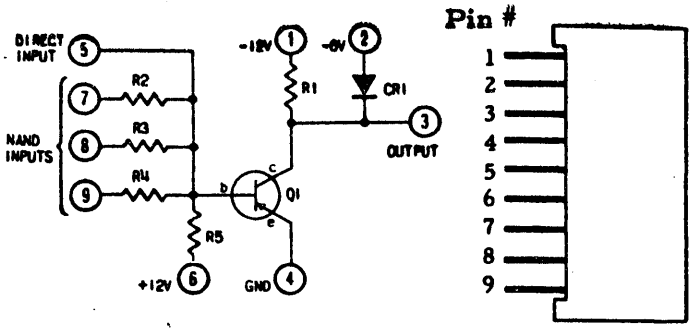


Figure 1 - EECO Universal Circuit Card

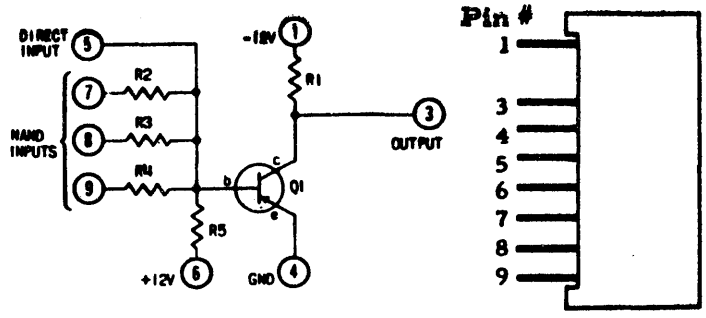
25 KC

3-input NAND/NOR

Q-411



Q-414



GENERAL

Q-411 and Q-414 are three input NAND/NOR circuits, based on resistor-transistor logic. Q-411 and Q-414 are identical circuits except Q-411 has a clamped output to provide uniform output voltage levels and Q-414 does not. There is a direct input provided in each unit for additional circuit versatility. These units function as NAND or as NOR logic according to the assigned logic level voltages:

For NAND:	"1" (true) = 0V "0" (false)= -6V	For NOR:	"1" (true) -6V "0" (false)= 0V
NAND output =	$\overline{ABC} = \overline{A} + \overline{B} + \overline{C}$	NOR output =	$\overline{A + B + C} = \overline{A}\,\overline{B}\,\overline{C}$

ELECTRICAL SPECIFICATIONS (NAND LOGIC)

		Min.	Max.	Units
Input:				
Frequency		0	25.0	KC
Rise Time	Q-411	---	2.0	µsec
	Q-414	---	2.0	µsec
Fall Time	Q-411	---	5.0	µsec
	Q-414	---	10.0	µsec
True Level		0	-0.5	volts
False Level		-6.0	-12.0	volts
Input Load		---	1.0	load units
Output:				
Rise Time	Q-411	---	2.0	µsec
	Q-414*	---	2.0	µsec
Fall Time	Q-411	---	5.0	µsec
	Q-414*	---	10.0	µsec
True Level		0	-0.5	volts
False Level	Q-411	-6.0	-6.5	volts
	Q-414	-6.0	-12.0	volts
True Level Delay		---	2.0	µsec
False Level Delay		---	3.5	µsec
Drive Capability		---	4.0	load units
Power Requirements:				
-12V, ±5%		---	3.0	ma
+12V, ±5%		---	0.1	ma
-6V, ±5%	(Q-411 only)	---	1.3	ma

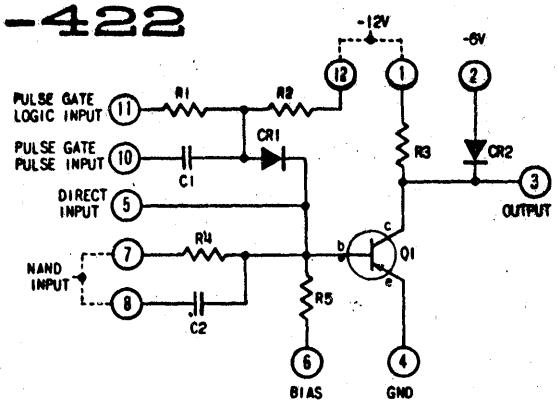
*Measured between 0VDC and -6VDC

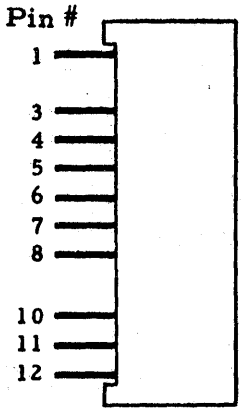
Figure 2 - Three input NAND/NOR

MultiCircuit

Q-422

100KC





GENERAL

Q-422 is a multi-purpose digital circuit used to produce flip-flops, one shots, multivibrator, squaring circuits, pulse amplifiers, etc.

Q-422 has a clamped output to provide uniform output voltage levels. When inputs to Q-422 come from unclamped circuits, care must be taken that the amplitude of the pulse gate pulse does not exceed the false level of the pulse gate logic input. For example, with a DC logic input of -8V, the pulse input must not exceed 8 volts in amplitude.

ELECTRICAL SPECIFICATIONS (NAND LOGIC)

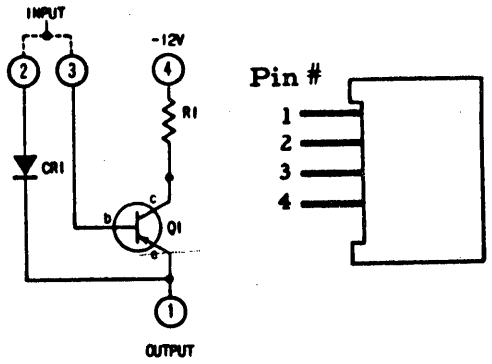
Input:	Min.	Max.	Units
Frequency:			
NAND	0	100.0	KC
Pulse Gate Logic	0	50.0	KC
Pulse Gate Pulse	0	100.0	KC
Rise Time:			
NAND	---	0.5	μsec
Pulse Gate Pulse	---	0.5	μsec
Fall Time:			
NAND	---	1.0	μsec
True Level			
NAND	0	-0.5	volts
Pulse Gate Logic	0	-0.5	volts
False Level:			
NAND	-6.0	-12.0	volts
Pulse Gate Logic	-6.0	-12.0	volts
Amplitude, Pulse Gate Pulse (Positive-going)	5.5	12.0	volts
Enable Time, Pulse Gate	0.5	4.0	μsec
Disable Time, Pulse Gate	0.5	4.0	μsec

Figure 3 - Multicircuit

continued

Power Driver

Q-413



GENERAL

Q-413 is used to increase the loading capabilities of the NAND/NOR units. Q-413 can drive up to 25 NAND/NOR inputs.

ELECTRICAL SPECIFICATIONS

	Min.	Max.	Units
Input:			
Frequency	0	25.0	KC
True Level	0	-0.5	volts
False Level	-6.0	-12.0	volts
Input Load	---	1	load units
Output:			
Drive Capability	---	25.0	load units
DC Levels	essentially equal to input DC levels		
Rise and Fall time	essentially equal to input rise and fall time		
Power Requirements:			
-12V, ±5%	---	15.0	ma

Figure 4 - Power Driver

The major disadvantage of the Q - Series is that it requires more soldering, but it offers the advantages of design flexibility and low cost.

General Specifications

The general specifications of the Digiklok are as follows:

1. The Digiklok shall keep real time to a digital reading accuracy of 0.1±0.05 seconds and shall be capable of reading out tenths of seconds (0-9), seconds (0-59) minutes (0-59), hours (0-23), and day of the year (0-365/6).

2. The Digiklok shall be compatible with and shall communicate with the 1620 through the 1622 Paper Tape Reader channels with minimum alterations to the 1620 system.

3. The Digiklok shall not require high speed logic due to the fact that it is based on a 60 cps time standard and that it will be read at about a 7 kcps rate.

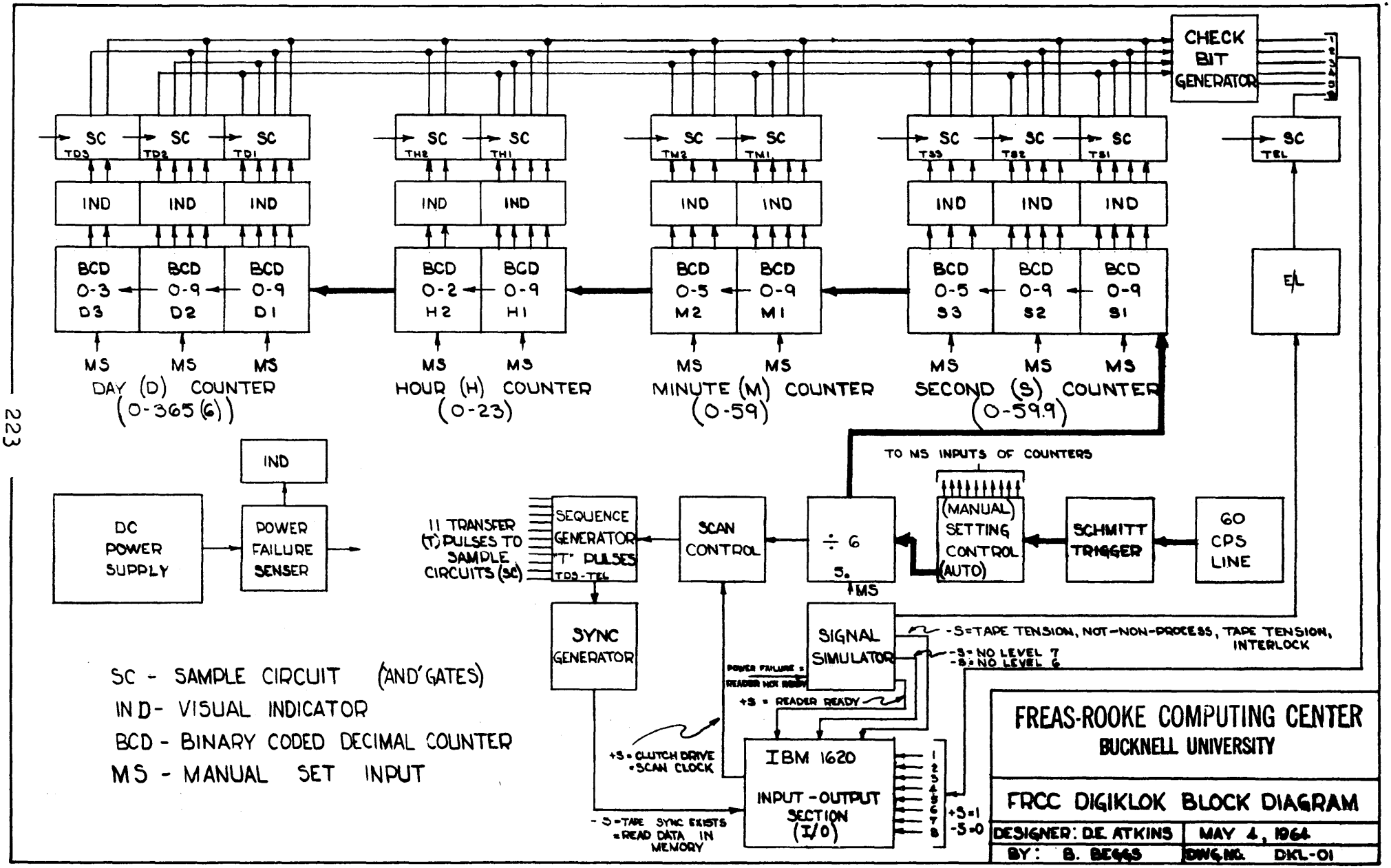
4. The Digiklok shall offer a BCD display.

General Description as Illustrated in Block Diagram, DKL-01

The general design and operation of the Digiklok is now explained with reference to the FRCC Digiklok Block Diagram, DWG. No. DKL-01. First consideration will be given to the operation of the time counters and second to the reading of the clock by the 1620. The logic flow for the counting circuits is indicated by the heavy lines, and that for the reading control and special functions by the lighter lines.

Time Base. Beginning with the functional block labeled 60 CPS LINE in the lower right side of DKL-01, note that it serves as the time base for the clock and that it drives the input of a Schmitt trigger which shapes the sinusoidal AC into a square wave of the same frequency. Although the frequency of the line will vary, it will be nearly constant when averaged over a day.

Binary Coded Decimals (BCD). The ten digit time number is counted and represented in the clock as a binary coded decimal in which each decimal number (0-9) is represented by its equivalent binary number as shown.


CHECK BIT GENERATOR
SC
IND
BCD
MS
DAY (D) COUNTER
(0-365 (6))
HOUR (H) COUNTER
(0-23)
MINUTE (M) COUNTER
(0-59)
SECOND (S) COUNTER
(0-59.9)
E/L
TO MS INPUTS OF COUNTERS
DC POWER SUPPLY
POWER FAILURE SENSER
11 TRANSFER (T) PULSES TO SAMPLE CIRCUITS (SC)
SEQUENCE GENERATOR
"T" PULSES
SCAN CONTROL
÷ 6
5 MS
(MANUAL) SETTING CONTROL (AUTO)
SCHMITT TRIGGER
60 CPS LINE
SYNC GENERATOR
SIGNAL SIMULATOR
-S=TAPE TENSION, NOT-NON-PROCESS, TAPE TENSION, INTERLOCK
-S= NO LEVEL 7
-S= NO LEVEL 6
+S = READER READY
+S = CLUTCH DRIVE
+SCAN CLOCK
IBM 1620
INPUT-OUTPUT SECTION (I/O)
+S=1
-S=0
-S=TAPE SYNC EXISTS
=READ DATA IN MEMORY
SC - SAMPLE CIRCUIT (AND GATES)
IND- VISUAL INDICATOR
BCD - BINARY CODED DECIMAL COUNTER
MS - MANUAL SET INPUT
FREAS-ROOKE COMPUTING CENTER
BUCKNELL UNIVERSITY
FRCC DIGIKLOK BLOCK DIAGRAM
DESIGNER: D.E. ATKINS
MAY 4, 1964
BY: B. BEGGS
DWG NO. DKL-01

Decimal number	BCD equivalent
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Table 2. Decimal to BCD Conversion

Each bit of the BCD is represented by the affirmative (1) output of a bistable multivibrator, commonly called a flip-flop. The group of flip-flops necessary to form one decimal number compose a BCD counter; a count of the maximum decimal number 9 requires four flip-flops. In the Digiklok, however, some counter maximums are less than 9, for example, the ten seconds counter counts to a maximum of 5 and the ten hours counter to a maximum of 2. These BCD counters therefore require only three and two flip-flops, respectively. In this report, unless otherwise defined, the term counter will refer to a BCD counter consisting of two, three, or four flip-flops.

Counters. With these terms in mind, continue following the counting logic on DKL-01. When the clock is operating in the normal automatic mode, the 60 pulse per second (pps) output of the Schmitt trigger drives through the Manual Setting Control into the input of the first counter (S_0) which divides the frequency of the signal by 6, i.e. the output triggers in the next counter if and only if S_0 has counted 6 input pulses representing 60/6 or 0.1 seconds. On the 6th pulse, S_0 resets to zero and begins recounting.

The output of counter S_0 drives the input of the tenth of second counter (S1) which immediately upon reaching a count of 10, resets to zero and triggers a count in the S2 counter. Likewise when the S2 counter reaches 9+1, it resets and triggers a count in the ten second counter which counts to 5 before resetting and triggering the one minute counter (M1) on the 5th pulse. The S counters thus effectively divide the 60 pps time base into sixth, tenth, unit and ten seconds.

Similarly the minute (M) counters count and register the output of the S counters, reset to zero after reaching a decimal count of M2=5 and M=9,

and transfer a count to the hour (H) counters. These counters reset to zero on the 24th pulse and transfer a count to the day (D) counters which total a maximum of 365 or 366 in a leap year.

Reading and Data Transfer. For the time to be read by the 1620, the outputs of the BCD counters must be gated into the computer input section. Data is read from the clock through the paper tape reader channels and thus the Digiklok operation must simulate that of the paper tape reader. Data transfer is by serial digits, parallel bits and is accomplished as now described.

Beginning at the large functional block labeled IBM 1620 INPUT OUTPUT SECTION, note the line from it to the Scan Control. When the 1620 is instructed to read the Digiklok and no transition is occurring as signaled by the connection from the S_0 counter to Scan Control, the Scan Control will be enabled by a level change. It in turn will enable the Sequence Generator which will produce a series of 11 transfer (T) pulses on 11 different lines, each connected to one of the Sample Circuits (SC) shown at the top of DKL-01.

The input to the SC function blocks consist of the transfer lines and the outputs of one BCD counter through a lamp indicator and appropriate driving circuits. When the T level is present on the input to a particular SC, the output of its counter is transferred to the data lines (parallel lines at top of drawing), through the Check Bit Generator, and into the 1620 Data Input Register. The Check Bit Generator maintains odd parity which is used for error detection, i.e., it produces an extra bit on line 5 if the sum of the "1" bits on line 1 to 4 is an even number (0=odd).

The output of the BCD counter selected by the Sequence Generator is now in the 1620 Data Input Register but the data is not transferred into core memory until a sync pulse is generated by the Digiklok Sync Pulse Generator and detected by the 1620. As indicated on the Block Diagram, the Sync Generator is driven by the Sequence Generator. The sync pulse, however, lags the T pulse time enough to insure that the inputs to the 1620 are stable before being read. Upon detecting the sync pulse, the 1620 transfers the data to core memory within 20usec and is ready to read data from the next SC enabled by the Sequence Generator.

Each time the 1620 is instructed to read the Digiklok, this reading sequence will be initiated at the D3 counter and continue serially through the End of Line signal (E/L). When the 1620 senses the E/L bit on line 8, it will negate the "1" signal to the Scan Control, the Sequence Generator will be inhibited, and data transfer will stop. The computer will proceed to execute the next programmed instruction.

Other Function Blocks. The Signal Simulator above the IBM function block must supply S levels to various inputs of the paper tape channels to

simulate control signals not used in the operation of the clock. "S" levels are IBM nomenclature and are defined as follows:

+S = -0.6v to -0.1v
-S = -12.48v to -6.87v

The power supply provides the proper levels and necessary current for the operation of the logic modules from a 115 VAC line. The Digiklok includes a relay to detect power failures which turns an indicator on, and negates the +S Reader Ready level from the Signal Simulator. The -S on this line will cause the computer to pause and indicate "Reader No Feed" on the console light display if the clock is addressed before being reset. The Reader Ready line is set to +S when the clock is manually set to the correct time.

Construction

Detailed designs were prepared during the summer of 1964 and construction began in September, 1964. The clock was essentially completed in February, 1965, and since then has been in continuous operation at the FRCC with no major failures or design errors detected.

The total cost for the project is summarized in Table 2.

ITEM	COST
EECO Q-Series Logic including hardware to construct 6 spare cards	$1358.61
Other hardware including that necessary for -12VDC and -6VDC Supply	$ 202.30
-12VDC Power Supply and Spare	$ 60.00
Cabinet	$ 29.07
Postage	$ 22.06
Telephone Expenses	$ 9.62
Labor	$ 700.00
TOTAL	$2381.66

TABLE 2 - Summary of Expenses,
Project Digiklok

III. THE AUTOLOG SOFTWARE

Introduction

In this section of the paper the Autolog System is described first from a users standpoint, secondly as a program modification to the Monitor I System, and lastly as a system for use with or without a digital clock. Program listings and the associated flow charts are included in the appendices. The step by step procedure for adding the Autolog Program to the Monitor I System can be found in Appendix B.

Before commencing with a discussion of the details of the Autolog System it might be appropriate to describe the entire system in general terms.

The Autolog System is used to generate a chronological table of jobs run under the Monitor I System. Each time a job is run a log record is entered into a log table area that has been reserved on the Monitor I disk pack. As presently operating, the table can handle as many as 400 entries before it must be dumped. The information in the table can be retrieved in either of two ways. The normal mode of retrieval is to run an edit and punch program that edits each log table entry, punches the edited entry on a card, and reinitializes the table so that it is ready to receive up to 400 new entries. An SPS listing of this edit and punch program is included as Appendix C. The second mode of retrieval is one that is not normally used, but rather is available so as to protect programs stored on disk by insuring that only 400 entries are made in the table. This routine is part of the Autolog program and with a minimum of operator attention will automatically dump the table to cards and initialize the table for 400 new entries. The system is designed in such a way as to prohibit running of a program if the log table is full.

An example of the output from the edit and punch routine is shown in Figure 5. The output from the edit and punch routine can now be used for any type of analysis desired. Figure 6 shows the output from one analysis program. This particular program condenses the output into a daily listing for each month.

Code No.	Machine	Date	Time On	Time Off	Net Time	Programmer
11176315	1620A	03 31 65	1350 279	1423 130	00 32 451	CONNER, T
11177211	1620A	03 31 65	1423 245	1425 265	00 02 020	DEFEO,BRENT
21109391	1620A	03 31 65	1425 406	1426 469	00 01 063	MORGAN
11177211	1620A	03 31 65	1426 573	1427 409	00 00 436	JANT
11177211	1620A	03 31 65	1427 552	1430 330	00 02 378	DEFEO,BRENT
11177211	1620A	03 31 65	1430 430	1431 266	00 00 436	HAUN
11177211	1620A	03 31 65	1431 412	1432 249	00 00 437	ZIMMERMAN
11171103	1620A	03 31 65	1432 393	1433 588	00 01 195	STOLL DAVID R.
11171103	1620A	03 31 65	1434 100	1436 282	00 02 182	STOLL DAVID R.
11177311	1620A	03 31 65	1436 392	1436 558	00 00 166	FRANTZ, L. I.
11174316	1620A	03 31 65	1437 069	1449 238	00 12 169	UHLER, ANDREW S
21109391	1620A	03 31 65	1449 348	1457 381	00 08 033	MORGAN
21109391	1620A	03 31 65	1457 485	1459 584	00 02 099	MORGAN
11177311	1620A	03 31 65	1500 098	1500 510	00 00 412	PRIESTER,PL
11176315	1620A	03 31 65	1501 027	1506 564	00 05 537	PANCZYSZYN,FRANK
11112308	1620A	03 31 65	1507 079	1513 287	00 06 208	RODNEY,P.F.
11174316	1620A	03 31 65	1513 407	1516 288	00 02 481	UHLER, ANDREW S
11177211	1620A	03 31 65	1517 107	1518 292	00 01 185	JANT
11177311	1620A	03 31 65	1519 564	1525 324	00 05 360	1FRANTZ, L. I.
11177211	1620A	03 31 65	1525 331	1527 173	00 01 442	WARNER
11177211	1620A	03 31 65	1528 278	1530 412	00 02 134	DEFEO,BRENT
11107393	1620A	03 31 65	1530 522	1533 094	00 02 172	DEFEO,BRENT
11107393	1620A	03 31 65	1533 260	1535 160	00 01 500	DEFEO,BRENT
21109391	1620A	03 31 65	1536 278	1537 345	00 01 067	MORGAN
11177211	1620A	03 31 65	1537 449	1539 034	00 01 185	JANT
21176719	1620A	03 31 65	1539 180	1539 565	00 00 385	WEBER G. L.
11177211	1620A	03 31 65	1540 228	1542 140	00 01 512	GELLER
21109391	1620A	03 31 65	1542 246	1543 311	00 01 065	MORGAN
11177211	1620A	03 31 65	1543 422	1546 066	00 02 244	CANTONI,JIM
11174316	1620A	03 31 65	1548 029	1552 094	00 04 065	UHLER, ANDREW S
11176344	1620A	03 31 65	1552 206	1555 171	00 02 565	STIDFOLE,R
11107393	1620A	03 31 65	1555 284	1558 257	00 02 573	DEFEO,BRENT
11177211	1620A	03 31 65	1558 556	1559 431	00 00 475	WARNER
12177001	1620A	03 31 65	1601 303	1605 286	00 03 583	GATSKI,R.L.
13177002	1620A	03 31 65	1605 450	1606 021	00 00 171	FRANTZ, L. I.
21109391	1620A	03 31 65	1606 281	1609 410	00 03 129	MORGAN
21109391	1620A	03 31 65	1609 516	1610 584	00 01 068	MORGAN
11177311	1620A	03 31 65	1611 110	1612 046	00 00 536	BERRIERJV 3-31-65
11176344	1620A	03 31 65	1616 064	1619 447	00 03 383	SMITH, DAVID F.
11177211	1620A	03 31 65	1620 238	1622 312	00 02 074	CANTONI,JIM
11176313	1620A	03 31 65	1630 491	1635 119	00 04 228	JOHN COX
13177002	1620A	03 31 65	1641 576	1642 591	00 01 015	FRANTZ, L. I.

Figure 5

AUTOLOG DAILY TIME REPORT
03 MONTH 1965

DAY	NO OF ENTRIES	TOTAL TIME HRS	MIN	SEC
1	110	9	9	45
2	28	11	16	0
3	105	7	20	27
4	107	5	21	38
5	64	3	57	22
6	38	1	25	51
7	45	3	46	17
8	105	6	37	49
9	142	12	7	59
10	142	20	37	58
11	206	16	42	25
12	227	17	30	27
13	152	4	59	22
15	198	12	56	24
16	68	12	39	18
17	24	6	41	32
18	216	12	37	58
19	166	14	58	55
20	120	7	26	21
21	127	11	24	16
22	233	16	58	25
23	204	9	50	31
24	151	11	52	44
25	204	13	59	34
26	149	12	37	5
27	65	6	58	37
28	27	1	38	57
29	124	8	6	21
30	122	9	43	48
31	95	10	30	58
TOTAL	3,764	301	55	16

FIGURE 6

Autolog from a Users Standpoint

In writing the Autolog modifications for the Monitor I System an effort was made to minimize the changes in the operating procedure of the Monitor I System. It was assumed that fewer operating errors would result if the deviation from normal Monitor operation was small. The only change made in the operating procedure was one that will help avoid confusion when using the multi-processing feature of Monitor. This change is not critical to the operation of the Autolog and could be easily deleted from the System.

To affect the use of the Autolog the user must simply fill in certain columns of the Monitor Job Card with the proper information. The twenty-eight columns of the JOB card normally used for comments, have been taken

```
‡ ‡ JOB                    11177001E. F. STAIANO..... TEST

                           Eight     Twenty Digit
                           Digit     Programmers
                           Numeric   Name Field
                           Job
                           Code
                           :      :                  :
                           :      :                  :
                           :      :                  :
                  COLUMN:  :      :                  :
                        31    40                    60
```

SAMPLE JOB CARD

over by the Autolog routine. These twenty eight columns are divided into two fields. (See sample JOB card). The first of these fields, which must contain only numeric information, is eight digits long. These eight digits are stored as part of the log record and can be used for charge codes, job classification, student numbers, etc. The second field, which is twenty digits long, long, can contain either alphabetic or numeric information; although the first character in the field must be alphabetic. This field should contain the programmer's name plus any auxiliary information the programmer wishes to record.

Both of the two fields mentioned above must be filled in properly. If either of these two fields is improperly coded or left blank the job is not processed, the operator is informed of the error and the supervisor goes into a JOB card search. The above information is all that is required for the Autolog when used in conjunction with the digital clock.

The change made in the operating procedure of the Monitor I System is designed to solve the problem of relating which answers belong to which program. Anyone who has used the multi-processing feature of the Monitor System has probably experienced the above program-answer puzzle.

This problem is easily solved by punching a header card and a trailer card for each program. As soon as the Monitor types the JOB card, the information that was punched in the programmer's name field is punched on a card. After the job has been completed and before the END OF JOB message is typed, two more cards are punched behind the output from the job. The first of these two cards will cause a page skip when the cards are listed on the IBM 407. The second card is a blank that makes it unnecessary to use the non-process run out button to get all of the output from a program.

The above modification has greatly simplified the problem of multi-processing with Monitor. The operator does not have to remove each job as it is completed and is thus free to prepare jobs for stacking and to watch for check stops caused by undefined variables in Fortran II programs. Since the programmer's name appearing on the job card is typed by the monitor system, the problem of associating error listings with the appropriate program is non-existant.

Incorporation of the Autolog Program into the Monitor I System

The Autolog Program is made up of three principal parts. The first part of the program is executed through a set of interrupt instructions placed in the JOB CARD PROCESSOR routine in the Monitor I Program. The second part of the system is the log table dump routine which is automatically called into action when the job table is full. The third part of the program is executed through a set of interrupt instructions placed in the END OF JOB PROCESSOR routine in the Monitor I program.

The entire Autolog program requires 32 sectors of storage on the disk. Flow chart #1 in Appendix A shows the various sections of the Autolog program and their locations in disk storage. The second block in the flow chart indicates the location of the sector that contains the disk control field for the current log table entry and the automatic table dump routine are common program segments, i.e., they are in core with the JOB PROCESSOR patch and the END OF JOB PROCESSOR patch.

The process of automatic logging is perhaps best explained by following through the processing of a Monitor Job.

As soon as a JOB card is sensed by the Monitor Control Record Analyzer routine control is passed to the JOB CARD PROCESSOR routine in Monitor. This routine analyzes the information contained on the JOB CARD and types this information with the aid of the TYPE A MONITOR CONTROL CARD routine. After the JOB card has been typed and before control is transferred from the JOB CARD PROCESSOR routine a set of patch instructions, inserted in this routine, reads in the Autolog program from disk storage and branches to the beginning of the program. This section of the Autolog routine first checks the appropriate Autolog fields read from the JOB card and then depending upon the validity of these fields either continues the logging operation or informs the operator of the presence of an error in the field or fields. Should either or both of the fields be improperly coded the Autolog routine sends the Monitor into a JOB card search. The program containing the improper card is not processed.

Assuming that the JOB card is properly filled out, the Autolog next decides whether or not the previous job processed ended through the END OF JOB routine. This checking feature was necessary to account for the cases when Monitor is destroyed in core and a cold start is required. Each time a program ends in this disastrous fashion an identifying record is placed in the table.

Using the information punched on the JOB card in columns 33 thru 60, the Autolog program reads the digital clock and stores on disk a JOB begun entry. This entry is made in the sector specified by the current log table entry disk control field. Upon completion of this task the Autolog executes the replaced Monitor instructions and transfers control back to the JOB PROCESSOR routine in Monitor.

No further interrupts are encountered until the job has been completed and Monitor branches to the END OF JOB PROCESSOR. At this point the second interrupt occurs and the second section of the Autolog program is read into core. After checking several indicators, the log record that was started back in the JOB PROCESSOR routine is read back into core. At this point the clock is interrogated again and the reading is placed in the log record which is in turn written back into the log table. The current log table entry disk control field sector address is then incremented by one and tested for size. If the table is full, control is transferred to the table dump routine. If the table is not full the replaced Monitor instructions are executed and control is returned to Monitor.

The automatic table dump routine does not edit the log table entries as they are punched on cards. This routine dumps the table exactly as it is stored on the disk. A separate program must be used to edit this output.

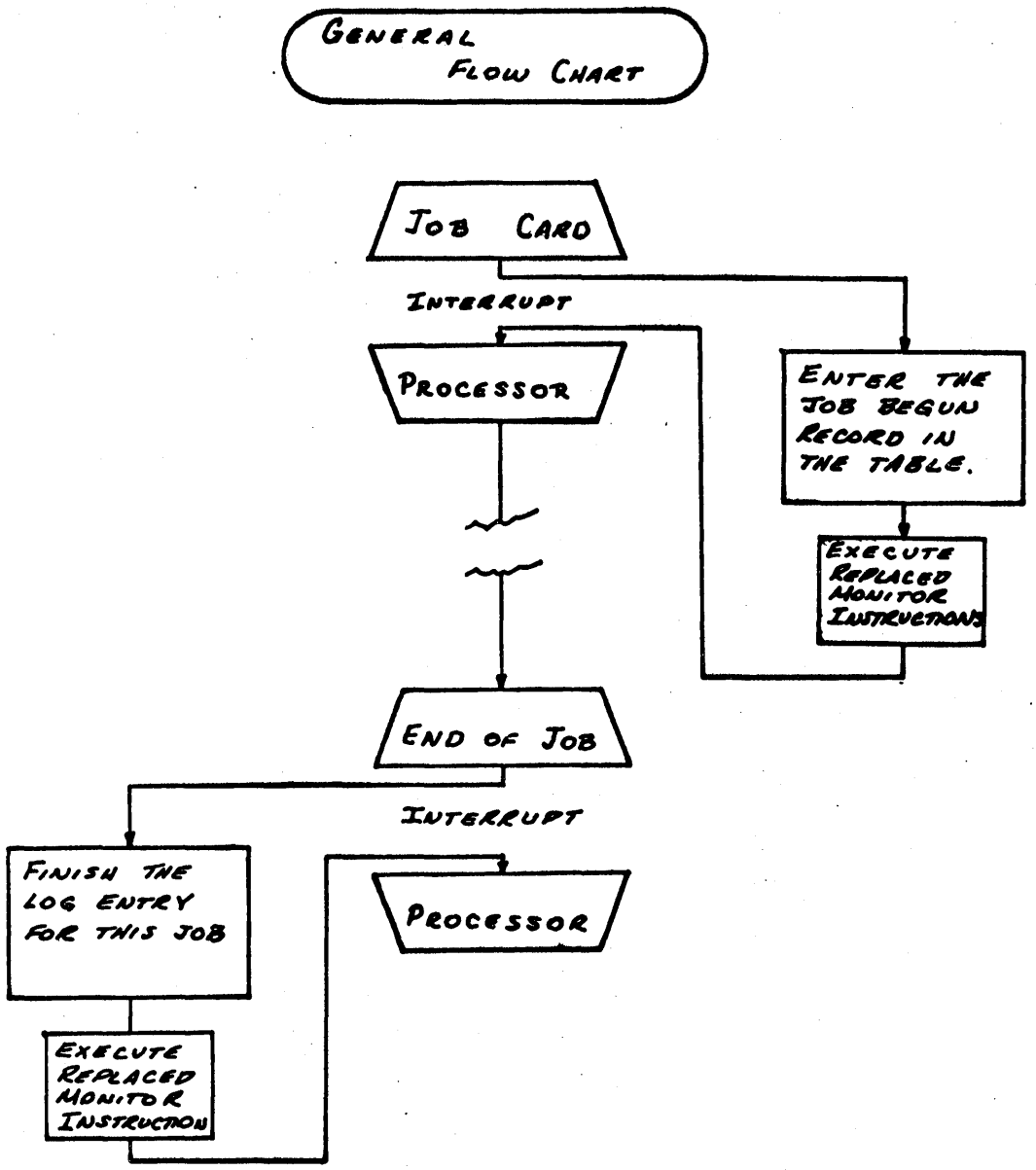

GENERAL
FLOW CHART
JOB CARD
INTERRUPT
PROCESSOR
ENTER THE JOB BEGUN RECORD IN THE TABLE.
EXECUTE REPLACED MONITOR INSTRUCTIONS
END OF JOB
INTERRUPT
FINISH THE LOG ENTRY FOR THIS JOB
PROCESSOR
EXECUTE REPLACED MONITOR INSTRUCTION

Figure 7

Should the above job have ended abnormally, the log record would have been completed by the JOB PROCESSOR interrupt program.

For further details of the Autolog program the reader is referred to Appendix A which contains flow charts and SPS listings of the program.

Special Features of the Autolog System

Operation without the digital clock. The Autolog Program can be used without a digital clock in either of two ways. The first way requires that the three Read Numeric Paper Tape instructions, used for reading the clock, be changed to Read Numeric Typewriter. This would require the operator to enter the time from the typewriter at the beginning and end of each job. This has the obvious disadvantage of slowing down the processing of programs while it has the advantage of forcing each user to log on and off the machine.

A second way to utilize the Autolog Program without a digital clock is to NOP the Read Numeric Paper Tape instructions and log only the programmer's name and the job code. This would provide a record of who used the machine and for what job but would not give any information concerning the amount of time used. This change would not increase the time necessary for logging and in addition would not require any operator intervention.

Compatability. The Autolog Program is compatible with FORGO-D. Certain features of FORGO-D are undesirable for automatic logging, however, and a further modification has been made to Monitor and FORGO which eliminates these disadvantages. These changes put FORGO on the same level as Fortran II, SPS II and the DUP routines. The double comment card has been eliminated in favor of a ‡ ‡ FORG control card. This card can be processed by existing Monitor Subroutines and results in a far simpler modification than FORGO-D. All halts have been removed from FORGO to allow for multiprocessing without operator intervention. This modification for FORGO can be used with or without the Autolog program.

PPAD 523 C
Rev. 2/65
Rev. 4/65

FORTRAN II INCREMENTAL PLOTTER SUBROUTINE
(DFIPS)

Pablo Larrea

Princeton-Pennsylvania Accelerator
Princeton University, Princeton, New Jersey

July 6, 1964

AEC Contract No. AT(30-1)-2137

FORTRAN II INCREMENTAL PLOTTER SUBROUTINE
(DFIPS)

Introduction

DFIPS is a general purpose routine for use in FORTRAN II programs for an IBM 1620 disk system with a Calcomp 565 Plotter. Its main function is to provide with a single routine and as few FORTRAN statements as possible a complete drawing.

Functions of Subroutine

The DFIPS subroutine will accept one pair of arrays at a time several of which can be included in one drawing. The first set of arrays for a drawing will have scaling performed on it, or it will give the scaling information; also the axes will be labelled as well as the drawing title will be drawn. Other sets of arrays will be superimposed on the same axes allowing 1/2" over the edges.

Subroutine Calls

There are 3 general types of calls to DFIPS, each of them accomplishes a specific task. The first type starts a new drawing with its labeling and scaling and draws the first points or curve. The second type superimposes on the axes drawn by the first call additional points or curves. The third type is for any additional labeling necessary beyond what is normally obtained by call type 1. Calls type 1 and 3 have at several places optional arguments which should be included only if they are necessary, since not all arguments have to be present at all times.

General Call (Type 1)

CALL DFIPS (X,Y,N,M,SX,XMIN,DELTX,YMIN,DELTY,R1,R2,R3) is the general call which starts a new drawing. The first two arguments are the names of floating point FORTRAN arrays. The successive elements of which contain the abscissas and ordinates respectively of the data points

to be plotted. The third argument (N) is a FORTRAN integer or integer variable which specifies the number of points to be plotted. The fourth argument (M) is a FORTRAN integer or integer variable whose 2 low order digits take the following meaning:

low order digit (units)
- if -1 draw line connecting points without drawing markers at the points
- 0 draw line connecting points and also mark the points
- 1 only draw marks without connecting the points

If the units position had an 0 or 1 then the tens position will be interrogated to determine which marker is to be used for the point according to the following:

0 = dot
1 = octagon
2 = plus sign
3 = triangle
4 = upside down triangle
5 = square
6 = diamond
7 = asterisk
8 = four pointed star
9 = star

The high order digit (thousands position) will take a meaning if the line is to be drawn (low order digit 0 or -1) and it will be: if 1 draw dashed lines connecting the points; if any other number draw full line.

Examples

If M = 30 the points will have triangles around them and be connected by a solid line.

If M = -1001 the points will be connected by a dashed line and no marker drawn.

If M = 1001 the points will be marked by a dot and no line will connect them. (NOTE: When a number is made negative the - sign goes with the low order digit.)

The next argument (SX) refers to the length of the X axis in inches. The Y axis is always fixed at 9 inches; and the X axis at whatever length is indicated by this argument which must be in floating point form and cannot exceed 99. inches. If it is not an integer number of inches its value will be chopped to integer. If it is desired to pre-select the scale, the next 4 arguments are entered; otherwise they are not included. The first of them (XMIN) represents the lowest value that will be printed on the X axis; the next is (DELTX) the number of units per inch along the X axis; the others are Y axis equivalents of them. (NOTE: If XMIN is some odd number such as 5121.3562 and DELTX 200. The values on the scale will represent 5121.3, 5321.3, etc., so care must be taken in selecting values of XMIN and YMIN, and not just the lowest values in the arrays.)

The last 3 sets of arguments (R1, R2 and R3) provide the drawing title, X axis and Y axis labels respectively.

Each of them can have from 1 to 11 arguments. They constitute essentially what would be found normally after a PRINT statement; that is a reference to a FORMAT statement number followed by a list of elements (if necessary) each of which is separated by a comma. In order to identify the FORMAT statement number it must be preceeded by a $ (dollar sign); so the indicator character ($) can be thought of as representing an imaginary WRITE ON PLOTTER type statement.

There are some important restrictions imposed by the compiler on this part

a) Maximum length of 80 characters of output.
b) FORMAT statement must have preceeded in the program its use.
c) No DO loops implied or otherwise allowed; but if (A(I), I=1,5) is desired it can be written as A(1), A(2), A(3), A(4), A(5) without any problem.
d) No carriage returns(/) can be given in the FORMAT statement; but the slash can be valid in a Hollerith string.

Note also that as written the program can handle only up to 11 arguments (a FORMAT statement number and 10 items on the list).

On writing a line all leading and trailing blanks are eliminated for speed.

TYPE 2 CALL

CALL DFIPS (X, Y, N, M)

This call is used when it is desired to fit other sets of data points in the same drawing. No scaling will be performed for these; they will only be fit on the same set of axes. A general call must have preceded this statement. If a point falls more than 1/2 inch outside of the limits of the drawing, a card will be punched containing the identification for the point, its coordinates and the message "out of range". In this call the X and Y have the same meaning as on the previous case; the N will contain the number of points; and M has the same meaning as before.

TYPE 3 CALL

CALL DFIPS (X, Y, N, R1)

This call is used when it is desired to add another line of writing at a specific point. It must be preceded by a general call.

The X and Y are floating point representation of the location where this title is to appear. If they are not given the data will be written where the pen is (at the end of a line to identify it). This line of data will normally be written in letters .18 inch high (size 3 since all sizes are multiples of .06) and oriented parallel to the X axis. The element N if present can modify both of these. Its value will indicate size number (if 1 letters will be .06 inches high, if 5 then .3 inches etc.), and its sign orientation; positive = parallel to X axis, negative = parallel to Y axis. Note: Only the last digit of size number will be used so maximum size possible is .54 inches. The last set of arguments being the data to be written. For an explanation see R1, R2 and R3 explanation in the general call.

RESULTS

The results will be a drawing which will have the axes drawn with tick marks of .1 inches in length every inch in each of the axes. At each tick mark will be the value on the scale at that point. If the values are between 1.0000 and 999.9999 no factoring will be performed on them; but otherwise they will have a multiplication factor associated with them. The numbers will consist of up to 4 decimal places with the trailing zeroes deleted for speed and neatness. These numbers indicating scale will be .06 inches in height and .04 inches in width with .02 inch separation between numbers. The units per inch used on the scale will be an integer or an integer times an appropriate power of ten (only 1, 2,4,5, & 8 are used, example .02 units/inch or 5000 units/inch) such that all the points in the first array are included in the drawing or as given by XMIN, DELTX, YMIN, DELTY. In the scaling process both for the X and Y axes their respective maximum and minimum values are obtained. Because only 4 places are output after the decimal point on the scales, the points should differ by at least 1 in their fourth most significant place. (Example: if X max = .00324658 X min should be $\leq$ .00324558 so that the scales show correct values.) The labels for the axes are with letters .18 inches high .12 inches wide with a space .06 inches wide

between letters. They start .6 inches away from the intersection of the axes. The drawing title will be below the X axis label in letters .24 inches high .16 inches wide and .08 inches apart. These dimensions are important to know the number of characters that fit in a particular size axis (always 1" allowance should be made in case scaling is performed so the scaling factor has space to go).

ERROR MESSAGES

Non Fatal:

If a point falls outside the normal drawing area (allowing 1/2 inch overlap) a card will be punched containing all information on the point

Fatal:

These are typed as ERROR DFIPS I meaning

I = 0 scaling is to be performed on a type 1 call and number of points on array (N) is less than 2

I = 1 scaling is to be performed on a type 1 call and on one of the axes its maximum and minimum values are equal

I = 2 line to be drawn with zero points on it

I = 3 wrong number of parameters in a call

I = 4 on call type 1 the length specified for X axis is > 99 inches or < 1. inch

On all of these program will halt and if start is pressed it will return to main program.

I=5 on call type 3 the size of character to be drawn has been specified as 0. Options available: a) Pressing start will neglect the title or b) Press insert, type 42 R-S will draw it size 3 with orientation parallel to X axis.

```
C
C       GENERAL PLOT PROGRAM
C
      DIMENSION X(100),Y(100),TITLE(10),XL(10),YL(10)
    9 FORMAT (E14.7,E14.7,I1,I1,I5)
   10 FORMAT (10A4)
   11 FORMAT (4E14.7)
    1 N=0
    2 READ 9,XP,YP,ICALL,IOP,M
      ICALL=ICALL+1
      GO TO (3,4,5,6,7),ICALL
    3 X(N+1)=XP
      Y(N+1)=YP
      N=N+1
      GO TO 2
    4 READ 11,SX
      READ 10,TITLE,XL,YL
      IF (IOP) 41,42,41
   41 READ 11,XMIN,DELTX,YMIN,DELTY
      CALL DFIPS(X,Y,N,M,SX,XMIN,DELTX,YMIN,DELTY,$10,TITLE(1),TITLE(2),
     1TITLE(3),TITLE(4),TITLE(5),TITLE(6),TITLE(7),TITLE(8),TITLE(9),
     2TITLE(10),$10,XL(1),XL(2),XL(3),XL(4),XL(5),XL(6),XL(7),XL(8),
     3XL(9),XL(10),$10,YL(1),YL(2),YL(3),YL(4),YL(5),YL(6),YL(7),YL(8)
     4,YL(9),YL(10))
      GO TO 1
   42 CALL DFIPS(X,Y,N,M,SX,                          $10,TITLE(1),TITLE(2),
     1TITLE(3),TITLE(4),TITLE(5),TITLE(6),TITLE(7),TITLE(8),TITLE(9),
     2TITLE(10),$10,XL(1),XL(2),XL(3),XL(4),XL(5),XL(6),XL(7),XL(8),
     3XL(9),XL(10),$10,YL(1),YL(2),YL(3),YL(4),YL(5),YL(6),YL(7),YL(8)
     4,YL(9),YL(10))
      GO TO 1
    5 CALL DFIPS(X,Y,N,M)
      GO TO 1
    6 READ 10,XL
      IF (IOP) 61,62,61
   61 READ 11,XP,YP
      CALL DFIPS(XP,YP,M,$10,XL(1),XL(2),XL(3),XL(4),XL(5),XL(6),XL(7),
     1XL(8),XL(9),XL(10))
      GO TO 1
   62 CALL DFIPS(      M,$10,XL(1),XL(2),XL(3),XL(4),XL(5),XL(6),XL(7),
     1XL(8),XL(9),XL(10))
      GO TO 1
    7 CALL EXIT
      END
0.0000000E-99 0.0000000E-99                                          111
2.0000000E-01 1.9866932E-01                                          211
4.0000000E-01 3.8941834E-01                                          311
6.0000000E-01 5.6464247E-01                                          411
8.0000000E-01 7.1735609E-01                                          511
1.0000000E+00 8.4147098E-01                                          611
1.2000000E+00 9.3203908E-01                                          711
1.4000000E+00 9.8544973E-01                                          811
1.6000000E+00 9.9957360E-01                                          911
1.8000000E+00 9.7384763E-01                                         1011
2.0000000E+00 9.0929742E-01                                         1111
2.2000000E+00 8.0849640E-01                                         1211
2.4000000E+00 6.7546318E-01                                         1311
2.6000000E+00 5.1550137E-01                                         1411
2.8000000E+00 3.3498815E-01                                         1511
3.0000000E+00 1.4112001E-01                                         1611
```

```
3.2000000E+00-5.8374140E-02                                          1711
3.4000000E+00-2.5554110E-01                                          1811
3.6000000E+00-4.4252044E-01                                          1911
3.8000000E+00-6.1185789E-01                                          2011
4.0000000E+00-7.5680249E-01                                          2111
4.2000000E+00-8.7157577E-01                                          2211
4.4000000E+00-9.5160207E-01                                          2311
4.6000000E+00-9.9369100E-01                                          2411
4.8000000E+00-9.9616460E-01                                          2511
5.0000000E+00-9.5892427E-01                                          2611
5.2000000E+00-8.8345465E-01                                          2711
5.4000000E+00-7.7276448E-01                                          2811
5.6000000E+00-6.3126663E-01                                          2911
5.8000000E+00-4.6460217E-01                                          3011
                              10-1001
    7.0
DRAWING NO. 1
    X
        F(X)
                              30    0
   SINE
0.0000000E-99 1.0000000E+00                                           112
2.0000000E-01 9.8006657E-01                                           212
4.0000000E-01 9.2106099E-01                                           312
6.0000000E-01 8.2533561E-01                                           412
8.0000000E-01 6.9670671E-01                                           512
1.0000000E+00 5.4030230E-01                                           612
1.2000000E+00 3.6235775E-01                                           712
1.4000000E+00 1.6996714E-01                                           812
1.6000000E+00-2.9199520E-02                                           912
1.8000000E+00-2.2720209E-01                                          1012
2.0000000E+00-4.1614683E-01                                          1112
2.2000000E+00-5.8850111E-01                                          1212
2.4000000E+00-7.3739371E-01                                          1312
2.6000000E+00-8.5688875E-01                                          1412
2.8000000E+00-9.4222234E-01                                          1512
3.0000000E+00-9.8999249E-01                                          1612
3.2000000E+00-9.9829477E-01                                          1712
3.4000000E+00-9.6679819E-01                                          1812
3.6000000E+00-8.9675841E-01                                          1912
3.8000000E+00-7.9096771E-01                                          2012
4.0000000E+00-6.5364362E-01                                          2112
4.2000000E+00-4.9026082E-01                                          2212
4.4000000E+00-3.0733287E-01                                          2312
4.6000000E+00-1.1215252E-01                                          2412
4.8000000E+00 8.7498980E-02                                          2512
5.0000000E+00 2.8366218E-01                                          2612
5.2000000E+00 4.6851667E-01                                          2712
5.4000000E+00 6.3469287E-01                                          2812
5.6000000E+00 7.7556587E-01                                          2912
5.8000000E+00 8.8551951E-01                                          3012
                              20 1000
                              30   -1
 COSINE
0.0000000E-99 0.0000000E-99                                           116
2.0000000E-01 1.7203253E-02                                           216
4.0000000E-01 4.3164576E-02                                           316
6.0000000E-01 6.1269942E-02                                           416
8.0000000E-01 6.8024589E-02                                           516
1.0000000E+00 6.3841255E-02                                           616
```

```
 1.2000000E+00 4.9813959E-02                                          716
 1.4000000E+00 2.6712890E-02                                          816
 1.6000000E+00-5.2156232E-03                                          916
 1.8000000E+00-4.6118163E-02                                         1016
 2.0000000E+00-9.6133782E-02                                         1116
 2.2000000E+00-1.5450408E-01                                         1216
 2.4000000E+00-2.1757276E-01                                         1316
 2.6000000E+00-2.7421677E-01                                         1416
 2.8000000E+00-2.9535598E-01                                         1516
 3.0000000E+00-2.0987799E-01                                         1616
 3.2000000E+00 1.1281171E-01                                         1716
 3.4000000E+00 3.2934359E-01                                         1816
 3.6000000E+00 3.8915125E-01                                         1916
 3.8000000E+00 3.7185877E-01                                         2016
 4.0000000E+00 3.1632622E-01                                         2116
 4.2000000E+00 2.4094434E-01                                         2216
 4.4000000E+00 1.5379238E-01                                         2316
 4.6000000E+00 5.7728242E-02                                         2416
 4.8000000E+00-4.6950622E-02                                         2516
 5.0000000E+00-1.6085775E-01                                         2616
 5.2000000E+00-2.8394344E-01                                         2716
 5.4000000E+00-4.1301108E-01                                         2816
 5.6000000E+00-5.3656497E-01                                         2916
 5.8000000E+00-6.2423665E-01                                         3016
                              20   91
 2.2619715E-03 1.9392431E-02                                         3099
                              10    1
   7.
NO DRAWING
X
F(X)
 0.0000000E-02 7.8539816E-01                                          124
 1.9866932E-01 6.9528399E-01                                          224
 3.8941834E-01 6.2384233E-01                                          324
 5.6464247E-01 5.6869125E-01                                          424
 7.1735609E-01 5.2729599E-01                                          524
 8.4147098E-01 4.9748714E-01                                          624
 9.3203908E-01 4.7761872E-01                                          724
 9.8544973E-01 4.6657470E-01                                          824
 9.9957360E-01 4.6373290E-01                                          924
 9.7384763E-01 4.6893333E-01                                         1024
 9.0929742E-01 4.8246883E-01                                         1124
 8.0849640E-01 5.0510184E-01                                         1224
 6.7546318E-01 5.3810003E-01                                         1324
 5.1550137E-01 5.8326687E-01                                         1424
 3.3498815E-01 6.4290585E-01                                         1524
 1.4112001E-01 7.1958391E-01                                         1624
-5.8374140E-02 7.5704642E-01                                         1724
-2.5554110E-01 6.7258438E-01                                         1824
-4.4252044E-01 6.0616860E-01                                         1924
-6.1185789E-01 5.5528613E-01                                         2024
-7.5680249E-01 5.1747657E-01                                         2124
-8.7157577E-01 4.9071672E-01                                         2224
-9.5160207E-01 4.7351797E-01                                         2324
-9.9589100E-01 4.6491260E-01                                         2424
-9.9616460E-01 4.6441587E-01                                         2524
-9.5892427E-01 4.7199980E-01                                         2624
-8.8345465E-01 4.8809158E-01                                         2724
-7.7276448E-01 5.1359704E-01                                         2824
-6.3126663E-01 5.4993848E-01                                         2924
```

```
-4.64460217E-01 5.9907470E-01                                          3024
                                10   31
    7.2458
DRAWING NO. 2
X
F(X)
 0.0000000E-99 1.0000000E+00                                            123
 1.9866932E-01 8.3425845E-01                                            223
 3.8941834E-01 7.1972565E-01                                            323
 5.6464247E-01 6.3912367E-01                                            423
 7.1735609E-01 5.8229045E-01                                            523
 8.4147098E-01 5.4304415E-01                                            623
 9.3203908E-01 5.1758789E-01                                            723
 9.8544973E-01 5.0366423E-01                                            823
 9.9957360E-01 5.0010662E-01                                            923
 9.7384763E-01 5.0662472E-01                                           1023
 9.0929742E-01 5.2375287E-01                                           1123
 8.0849640E-01 5.5294552E-01                                           1223
 6.7546318E-01 5.9684990E-01                                           1323
 5.1550137E-01 6.5984766E-01                                           1423
 3.3498815E-01 7.4907034E-01                                           1523
 1.4112001E-01 8.7633202E-01                                           1623
-5.8374140E-02 9.4484549E-01                                           1723
-2.5554110E-01 7.9646934E-01                                           1823
-4.4252044E-01 6.9323109E-01                                           1923
-6.1185789E-01 6.2040212E-01                                           2023
-7.5680249E-01 5.6921598E-01                                           2123
-8.7157577E-01 5.3430913E-01                                           2223
-9.5160207E-01 5.1239955E-01                                           2323
-9.9369100E-01 5.0158224E-01                                           2423
-9.9616460E-01 5.0096069E-01                                           2523
-9.5892427E-01 5.1048427E-01                                           2623
-8.8345465E-01 5.3093926E-01                                           2723
-7.7276448E-01 5.6409075E-01                                           2823
-6.3126663E-01 6.1302058E-01                                           2923
-4.6460217E-01 6.8277930E-01                                           3023
                                20   910
 0.0000000E-99 8.8622692E-01                                            125
 1.9866932E-01 7.6160786E-01                                            225
 3.8941834E-01 6.7007113E-01                                            325
 5.6464247E-01 6.0287978E-01                                            425
 7.1735609E-01 5.5411136E-01                                            525
 8.4147098E-01 5.1976675E-01                                            625
 9.3203908E-01 4.9720183E-01                                            725
 9.8544973E-01 4.8476487E-01                                            825
 9.9957360E-01 4.8157646E-01                                            925
 9.7384763E-01 4.8741482E-01                                           1025
 9.0929742E-01 5.0268720E-01                                           1125
 8.0849640E-01 5.2848253E-01                                           1225
 6.7546318E-01 5.6671416E-01                                           1325
 5.1550137E-01 6.2037671E-01                                           1425
 3.3498815E-01 6.9396087E-01                                           1525
 1.4112001E-01 7.9409975E-01                                           1625
-5.8374140E-02 8.4574930E-01                                           1725
-2.5554110E-01 7.3191039E-01                                           1825
-4.4252044E-01 6.4823985E-01                                           1925
-6.1185789E-01 5.8694181E-01                                           2025
-7.5680249E-01 5.4273007E-01                                           2125
-8.7157577E-01 5.1204923E-01                                           2225
-9.5160207E-01 4.9257526E-01                                           2325
```

```
-9.9369100E-01 4.8289947E-01                                          2425
-9.9616460E-01 4.8234229E-01                                          2525
-9.5892427E-01 4.9086502E-01                                          2625
-8.8345465E-01 5.0906480E-01                                          2725
-7.7276448E-01 5.3825210E-01                                          2825
-6.3126663E-01 5.8062345E-01                                          2925
-4.6460217E-01 6.3955906E-01                                          3025
                               20   71
                               20    1
 3.0000000E+02 2.6179938E-03                                           188
 3.0002062E+02 2.3645884E-03                                           288
 3.0005997E+02 2.2572434E-03                                           388
 3.0009586E+02 2.2960745E-03                                           488
 3.0011682E+02 2.5218203E-03                                           588
 3.0011756E+02 3.0679880E-03                                           688
 3.0009624E+02 4.3922117E-03                                           788
 3.0005303E+02 9.1486751E-03                                           888
 2.9998958E+02-5.2940254E-02                                           988
 2.9990897E+02-6.8819154E-03                                          1088
 2.9981646E+02-3.8669381E-03                                          1188
 2.9972058E+02-2.8636181E-03                                          1288
 2.9963547E+02-2.4354003E-03                                          1388
 2.9958443E+02-2.2720801E-03                                          1488
 2.9960571E+02-2.2774240E-03                                          1588
 2.9976051E+02-2.4247955E-03                                          1688
 3.0011939E+02-2.5267930E-03                                          1788
 3.0041350E+02-2.3157491E-03                                          1888
 3.0056135E+02-2.2489761E-03                                          1988
 3.0059938E+02-2.3354468E-03                                          2088
 3.0055572E+02-2.6340540E-03                                          2188
 3.0045094E+02-3.3314255E-03                                          2288
 3.0030014E+02-5.1306444E-03                                          2388
 3.0011509E+02-1.3812568E-02                                          2488
 2.9990628E+02 1.7697768E-02                                          2588
 2.9968490E+02 5.5523331E-03                                          2688
 2.9946521E+02 3.4788035E-03                                          2788
 2.9926783E+02 2.7039513E-03                                          2888
 2.9912456E+02 2.3705185E-03                                          2988
 2.9908575E+02 2.2619715E-03                                          3088
                               11  320
    -7.2587
DRAWING NO. 3
X
F(X)
      299.        .25            -.06             .01
                               31    2
EXTRA LABEL UP HERE
     1.             8.5
                               31   -3
IT'S TIGHT IN HERE
    6.6          1.
 3.0000000E+02 0.0000000E-99                                           189
 3.0002062E+02 6.6662084E-04                                           289
 3.0005997E+02 1.3330668E-03                                           389
 3.0009586E+02 1.9993611E-03                                           489
 3.0011682E+02 2.6656286E-03                                           589
 3.0011756E+02 3.3320276E-03                                           689
 3.0009624E+02 3.9987172E-03                                           789
 3.0005303E+02 4.6658419E-03                                           889
 2.9998958E+02 5.3335185E-03                                           989
```

```
 2.9990897E+02 6.0018211E-03                                          1089
 2.9981646E+02 6.6707478E-03                                          1189
 2.9972058E+02 7.3401699E-03                                          1289
 2.9963547E+02 8.0097326E-03                                          1389
 2.9958443E+02 8.6786886E-03                                          1489
 2.9960571E+02 9.3456162E-03                                          1589
 2.9976051E+02 1.0007989E-02                                          1689
 3.0011939E+02 1.0662423E-02                                          1789
 3.0041350E+02 1.1317733E-02                                          1889
 3.0056135E+02 1.1977587E-02                                          1989
 3.0059938E+02 1.2641409E-02                                          2089
 3.0055572E+02 1.3308680E-02                                          2189
 3.0045094E+02 1.3978987E-02                                          2289
 3.0030014E+02 1.4652007E-02                                          2389
 3.0011509E+02 1.5327453E-02                                          2489
 2.9990628E+02 1.6004999E-02                                          2589
 2.9968490E+02 1.6684190E-02                                          2689
 2.9946521E+02 1.7364287E-02                                          2789
 2.9926783E+02 1.8044037E-02                                          2889
 2.9912456E+02 1.8721297E-02                                          2989
 2.9908575E+02 1.9392431E-02                                          3089
                          20    40
 7.8539816E-01 8.8622692E-01                                           155
 6.9528399E-01 7.6160786E-01                                           255
 6.2384233E-01 6.7007113E-01                                           355
 5.6869125E-01 6.0287978E-01                                           455
 5.2729599E-01 5.5411136E-01                                           555
 4.9748714E-01 5.1976675E-01                                           655
 4.7761872E-01 4.9720183E-01                                           755
 4.6657470E-01 4.8476487E-01                                           855
 4.6573290E-01 4.8157646E-01                                           955
 4.6893333E-01 4.8741482E-01                                          1055
 4.8246883E-01 5.0268720E-01                                          1155
 5.0510184E-01 5.2848253E-01                                          1255
 5.3810003E-01 5.6671416E-01                                          1355
 5.8326687E-01 6.2037671E-01                                          1455
 6.4290585E-01 6.9396087E-01                                          1555
 7.1958391E-01 7.9409975E-01                                          1655
 7.5704642E-01 8.4574930E-01                                          1755
 6.7258438E-01 7.3191039E-01                                          1855
 6.0616860E-01 6.4823985E-01                                          1955
 5.5528613E-01 5.8694181E-01                                          2055
 5.1747657E-01 5.4273007E-01                                          2155
 4.9071672E-01 5.1204923E-01                                          2255
 4.7351797E-01 4.9257526E-01                                          2355
 4.6491260E-01 4.8289947E-01                                          2455
 4.6441587E-01 4.8234229E-01                                          2555
 4.7199980E-01 4.9086502E-01                                          2655
 4.8809158E-01 5.0906480E-01                                          2755
 5.1259704E-01 5.3825210E-01                                          2855
 5.4993848E-01 5.8062345E-01                                          2955
 5.9907470E-01 6.3955906E-01                                          3055
                          10     1
    101.
ANOTHI R ERROR
X
F(X)
 7.8539816E-01 0.0000000E-99                                           156
 6.9528399E-01 1.7203253E-02                                           256
 6.2384233E-01 4.3164576E-02                                           356
```

```
 5.6869125E-01 6.1269942E-02                                          456
 5.2729599E-01 6.8024589E-02                                          556
 4.9748714E-01 6.3841255E-02                                          656
 4.7761872E-01 4.9813959E-02                                          756
 4.6657470E-01 2.6712890E-02                                          856
 4.6373290E-01-5.2156232E-03                                          956
 4.6893333E-01-4.6118163E-02                                         1056
 4.8246883E-01-9.6133782E-02                                         1156
 5.0510184E-01-1.5450408E-01                                         1256
 5.3810003E-01-2.1757276E-01                                         1356
 5.8326687E-01-2.7421677E-01                                         1456
 6.4290585E-01-2.9535598E-01                                         1556
 7.1958391E-01-2.0987799E-01                                         1656
 7.5704642E-01 1.1281171E-01                                         1756
 6.7258438E-01 3.2934359E-01                                         1856
 6.0616860E-01 3.8915125E-01                                         1956
 5.5528613E-01 3.7185877E-01                                         2056
 5.1747657E-01 3.1632622E-01                                         2156
 4.9071672E-01 2.4094434E-01                                         2256
 4.7351797E-01 1.5379238E-01                                         2356
 4.6491260E-01 5.7728242E-02                                         2456
 4.6441587E-01-4.6950622E-02                                         2556
 4.7199980E-01-1.6085775E-01                                         2656
 4.9809158E-01-2.8394344E-01                                         2756
 5.3597046-01-4.1301108E-01                                          2856
 5.4923848E-01-5.3666497E-01                                         2956
 5.9907470E-01-6.2423665E-01                                         3056
                              10    30
      .8
AND ANOTHER ONE
X
F(X)
 0.0000000E-99 3.0000000E+02                                          117
 0.0000000E-99 1.0000000E+00                                          113
                              10    40
    10.
THIS MAKES 3 IN A ROW
X
F(X)
 0.0000000E-99 1.0000000E+00                                          122
 1.9866932E-01 9.8006657E-01                                          222
 3.8941834E-01 9.2106099E-01                                          322
 5.6464247E-01 8.2533561E-01                                          422
 7.1735609E-01 6.9670671E-01                                          522
 8.4147098E-01 5.4030230E-01                                          622
 9.3203908E-01 3.6235775E-01                                          722
 9.8544973E-01 1.6996714E-01                                          822
 9.9957360E-01-2.9199520E-02                                          922
 9.7384763E-01-2.2720209E-01                                         1022
 9.0929742E-01-4.1614683E-01                                         1122
 8.0849640E-01-5.8850111E-01                                         1222
 6.7546318E-01-7.3739371E-01                                         1322
 5.1550137E-01-8.5688875E-01                                         1422
 3.3498815E-01-9.4222234E-01                                         1522
 1.4112001E-01-9.8999249E-01                                         1622
-9.8374140E-02-9.9829477E-01                                         1722
-2.5554110E-01-9.6679819E-01                                         1822
-4.4252044E-01-8.9675841E-01                                         1922
-6.1185789E-01-7.9096771E-01                                         2022
-7.5680249E-01-6.5364362E-01                                         2122
```

```
-8.7157577E-01-4.9026082E-01                                        2222
-9.5160207E-01-3.0733287E-01                                        2322
-9.9369100E-01-1.1215252E-01                                        2422
-9.9616460E-01 8.7498980E-02                                        2522
-9.5892427E-01 2.8366218E-01                                        2622
-8.8345465E-01 4.6851667E-01                                        2722
-7.7276448E-01 6.3469287E-01                                        2822
-6.3126663E-01 7.7556587E-01                                        2922
-4.6460217E-01 8.8551951E-01                                        3022
                                10    -1
   7.
DRAWING NO. 4
X
F(X)
 0.0000000E-99 0.0000000E-99                                         126
 1.9866932E-01 1.7203253E-02                                         226
 3.8941834E-01 4.3164576E-02                                         326
 5.6464247E-01 6.1269942E-02                                         426
 7.1735609E-01 6.8024589E-02                                         526
 8.4147098E-01 6.3841255E-02                                         626
 9.3203908E-01 4.9813959E-02                                         726
 9.8544973E-01 2.6712890E-02                                         826
 9.9957360E-01-5.2156232E-03                                         926
 9.7384763E-01-4.6118163E-02                                        1026
 9.0929742E-01-9.6133782E-02                                        1126
 8.0849640E-01-1.5450408E-01                                        1226
 6.7546318E-01-2.1757276E-01                                        1326
 5.1550137E-01-2.7421677E-01                                        1426
 3.3408815E-01-2.9535598E-01                                        1526
 1.4112001E-01-2.0987799E-01                                        1626
-5.8374140E-02 1.1281171E-01                                        1726
-2.5554110E-01 3.2934359E-01                                        1826
-4.4252044E-01 3.8915125E-01                                        1926
-6.1185789E-01 3.7185877E-01                                        2026
-7.5680249E-01 3.1632622E-01                                        2126
-8.7157577E-01 2.4094434E-01                                        2226
-9.5160207E-01 1.5379238E-01                                        2326
-9.9369100E-01 5.7728242E-02                                        2426
-9.9616460E-01-4.6950622E-02                                        2526
-9.5892427E-01-1.6085775E-01                                        2626
-8.8345465E-01-2.8394344E-01                                        2726
-7.7276448E-01-4.1301108E-01                                        2826
-6.3126663E-01-5.3666497E-01                                        2926
-4.6460217E-01-6.2423665E-01                                        3026
                                20    61
 1.0000000E+00 8.8622692E-01                                         135
 9.8006657E-01 7.6160786E-01                                         235
 9.2106099E-01 6.7007113E-01                                         335
 8.2533561E-01 6.0287978E-01                                         435
 6.9670671E-01 5.5411136E-01                                         535
 5.4030230E-01 5.1976675E-01                                         635
 3.6235775E-01 4.9720183E-01                                         735
 1.6996714E-01 4.8476487E-01                                         835
-2.9199520E-02 4.8157646E-01                                         935
-2.2720209E-01 4.8741482E-01                                        1035
-4.1614683E-01 5.0268720E-01                                        1135
-5.8850111E-01 5.2848253E-01                                        1235
-7.3739371E-01 5.6671416E-01                                        1335
-8.5688875E-01 6.2037671E-01                                        1435
-9.4222234E-01 6.9396087E-01                                        1535
```

```
-8.8999249E-01  7.9409975E-01                                          1635
-9.9829477E-01  8.4574930E-01                                          1735
-9.6679819E-01  7.8191039E-01                                          1835
-8.9675841E-01  6.4823985E-01                                          1935
-7.9096771E-01  5.8694181E-01                                          2035
-6.5364362E-01  5.4273007E-01                                          2135
-4.9026082E-01  5.1204923E-01                                          2235
-3.0733287E-01  4.9257526E-01                                          2335
-1.1215252E-01  4.8289947E-01                                          2435
 8.7498980E-02  4.8234229E-01                                          2535
 2.8366218E-01  4.9086502E-01                                          2635
 4.6851667E-01  5.0906480E-01                                          2735
 6.3469287E-01  5.3825210E-01                                          2835
 7.7556587E-01  5.8062345E-01                                          2935
 8.8551951E-01  6.3955906E-01                                          3035
                                20    80
 1.0000000E+00  1.0000000E+00                                           133
 9.8006657E-01  8.3425845E-01                                           233
 9.2106099E-01  7.1972565E-01                                           333
 8.2533561E-01  6.3912367E-01                                           433
 6.9670571E-01  5.8229045E-01                                           533
 5.4630230E-01  5.4304415E-01                                           633
 3.6235775E-01  5.1758789E-01                                           733
 1.6996714E-01  5.0366423E-01                                           833
-2.9199520E-02  5.0010662E-01                                           933
-2.2720209E-01  5.0662472E-01                                          1033
-4.1614683E-01  5.2375287E-01                                          1133
-5.8650111E-01  5.5294552E-01                                          1233
-7.3739371E-01  5.9684990E-01                                          1333
-8.5688875E-01  6.5984766E-01                                          1433
-9.4222234E-01  7.4907034E-01                                          1533
-9.8999249E-01  8.7633202E-01                                          1633
-9.9829477E-01  9.4484549E-01                                          1733
-9.6679819E-01  7.9646934E-01                                          1833
-8.9675841E-01  6.9323109E-01                                          1933
-7.9096771E-01  6.2040212E-01                                          2033
-6.5364362E-01  5.6921598E-01                                          2133
-4.9026082E-01  5.3430913E-01                                          2233
-3.0733287E-01  5.1239955E-01                                          2333
-1.1215252E-01  5.0158224E-01                                          2433
 8.7498980E-02  5.0096069E-01                                          2533
 2.8366218E-01  5.1048427E-01                                          2633
 4.6851667E-01  5.3093926E-01                                          2733
 6.3469287E-01  5.6409075E-01                                          2833
 7.7556587E-01  6.1302058E-01                                          2933
 8.8551951E-01  6.8277930E-01                                          3033
                                20    50
                                40    00
```

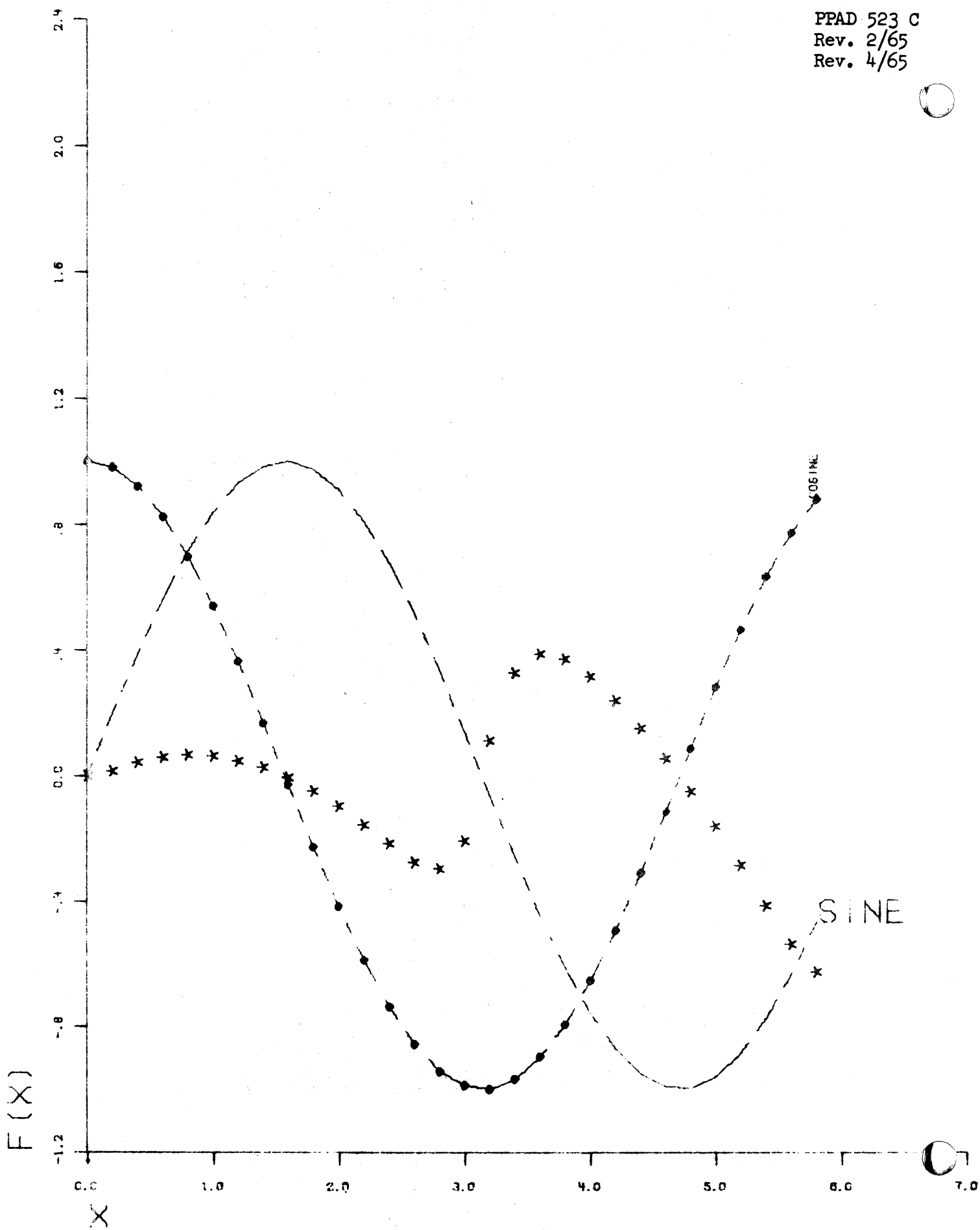
RESULTS

```
DRAWING    2 LINE   2 POINT    1 X= 0.0000000E-99 Y= 1.0000000E+00 OUT OF RANGE
DRAWING    2 LINE   2 POINT    2 X= 1.9866932E-01 Y= 8.3425845E-01 OUT OF RANGE
DRAWING    2 LINE   2 POINT   16 X= 1.4112001E-01 Y= 8.7633202E-01 OUT OF RANGE
DRAWING    2 LINE   2 POINT   17 X=-5.8374140E-02 Y= 9.4484549E-01 OUT OF RANGE
DRAWING    2 LINE   3 POINT    1 X= 0.0000000E-99 Y= 8.8622692E-01 OUT OF RANGE
DRAWING    2 LINE   3 POINT   17 X=-5.8374140E-02 Y= 8.4574930E-01 OUT OF RANGE
```

```
‡‡JOB

‡‡DUP

*DELETDFIPS

END OF JOB

‡‡JOB

‡‡SPS

*ASSEMBLE RELOCATABLE
*STORE RELOADABLE
*NAME DFIPS


END OF ASSEMBLY.
Ø846Ø CORE POSITIONS REQUIRED PLUS RELOCATION INCREMENT
ØØ784 STATEMENTS PROCESSED


DK LOADED DFIPS  Ø171 1Ø52ØØT129999ØØØØØ‡

‡‡FORX

*FANDKØ8Ø4
Ø4Ø6Ø CORES USED
59995 NEXT COMMON
END OF COMPILATION
EXECUTION


MAIN T4ØØØ Ø4Ø6Ø  LOADED
DFIPS  T8Ø6Ø Ø8458  LOADED
ERROR DFIPS 5
42RS
ERROR DFIPS Ø
ERROR DFIPS 2
ERROR DFIPS 4
ERROR DFIPS 4
ERROR DFIPS 1
END OF JOB
```

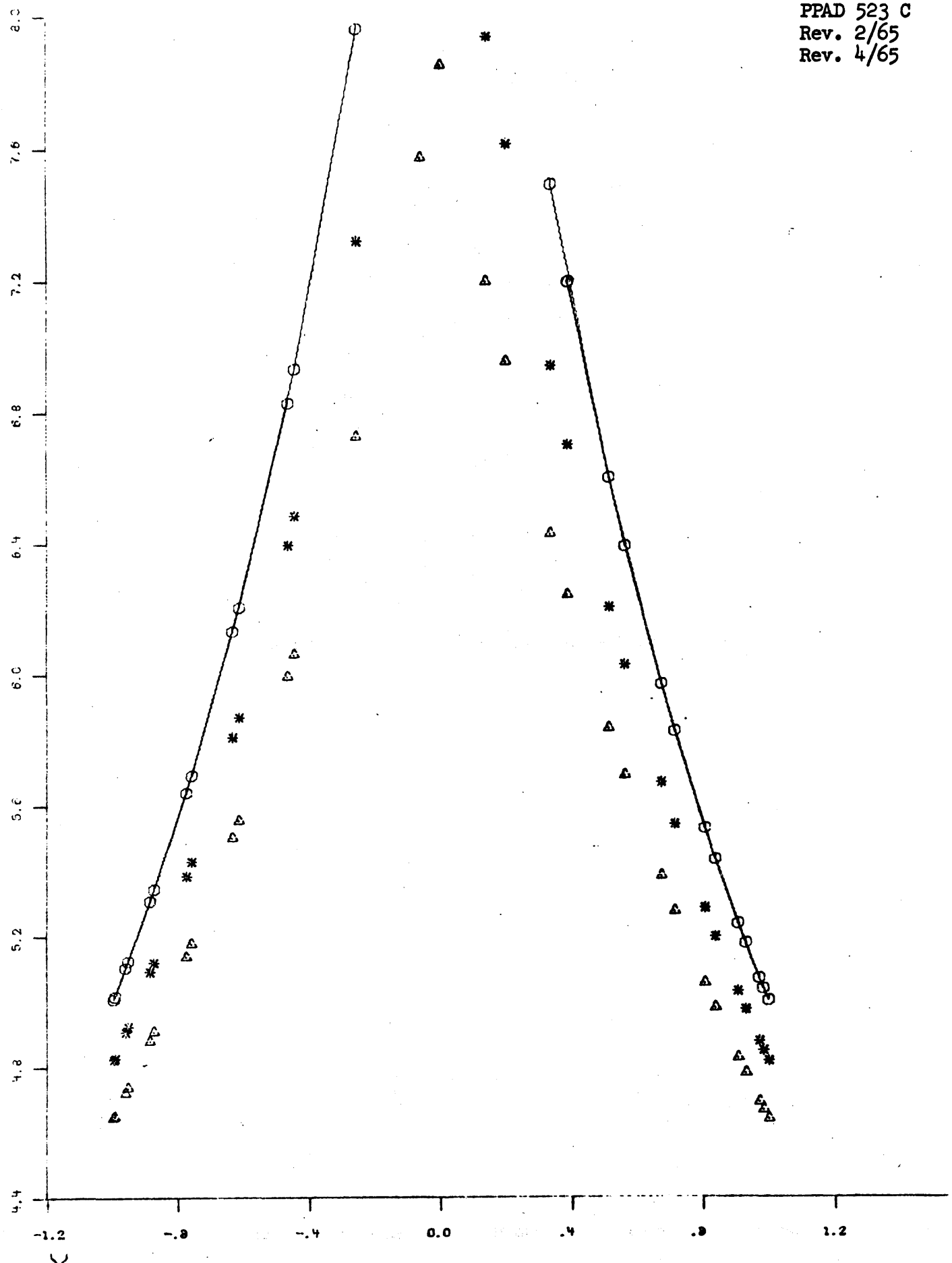
10 MIN. 40 SEC.

PPAD 523 C
Rev. 2/65
Rev. 4/65



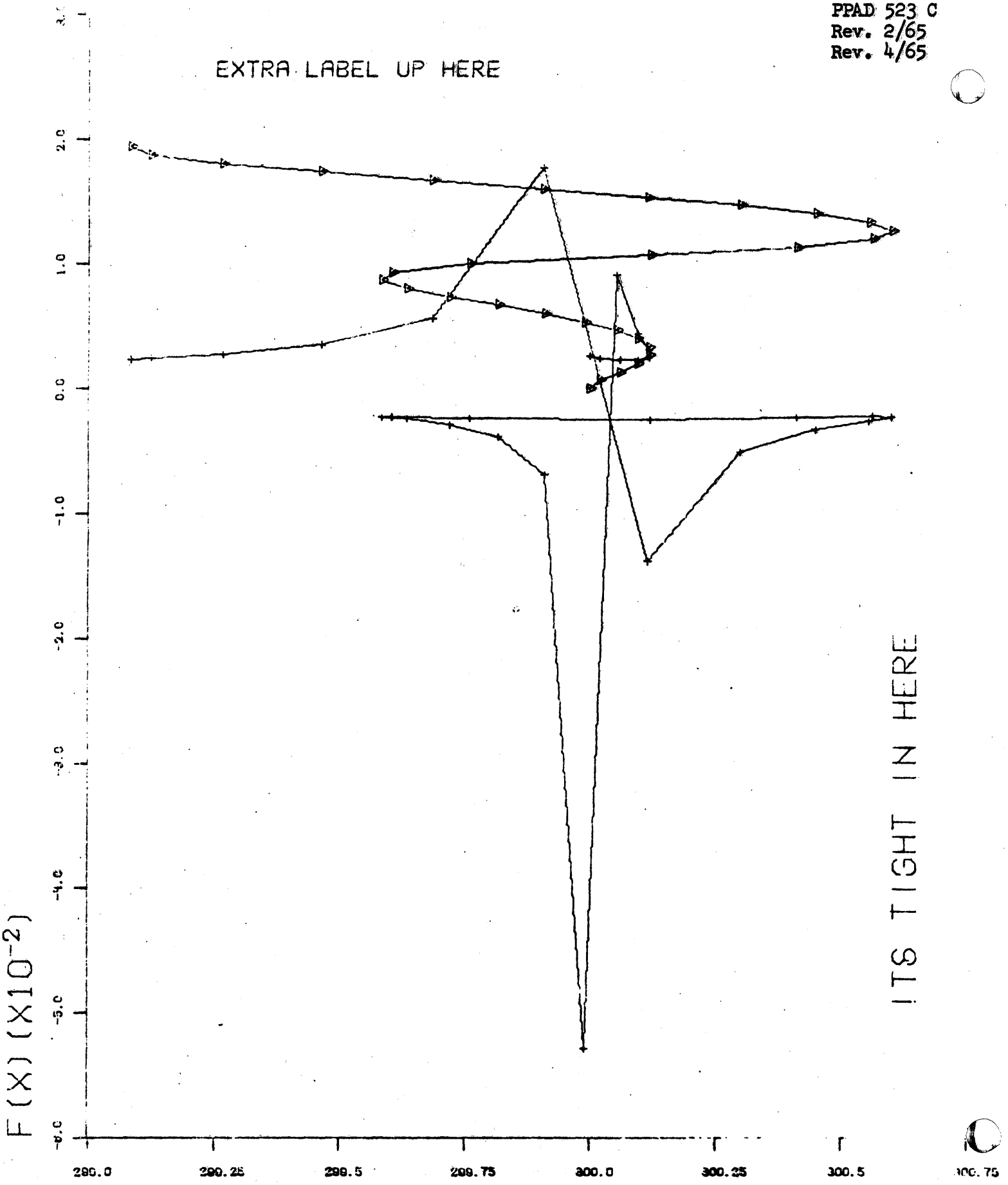
DRAWING NO. 1

PPAD 523 C
Rev. 2/65
Rev. 4/65



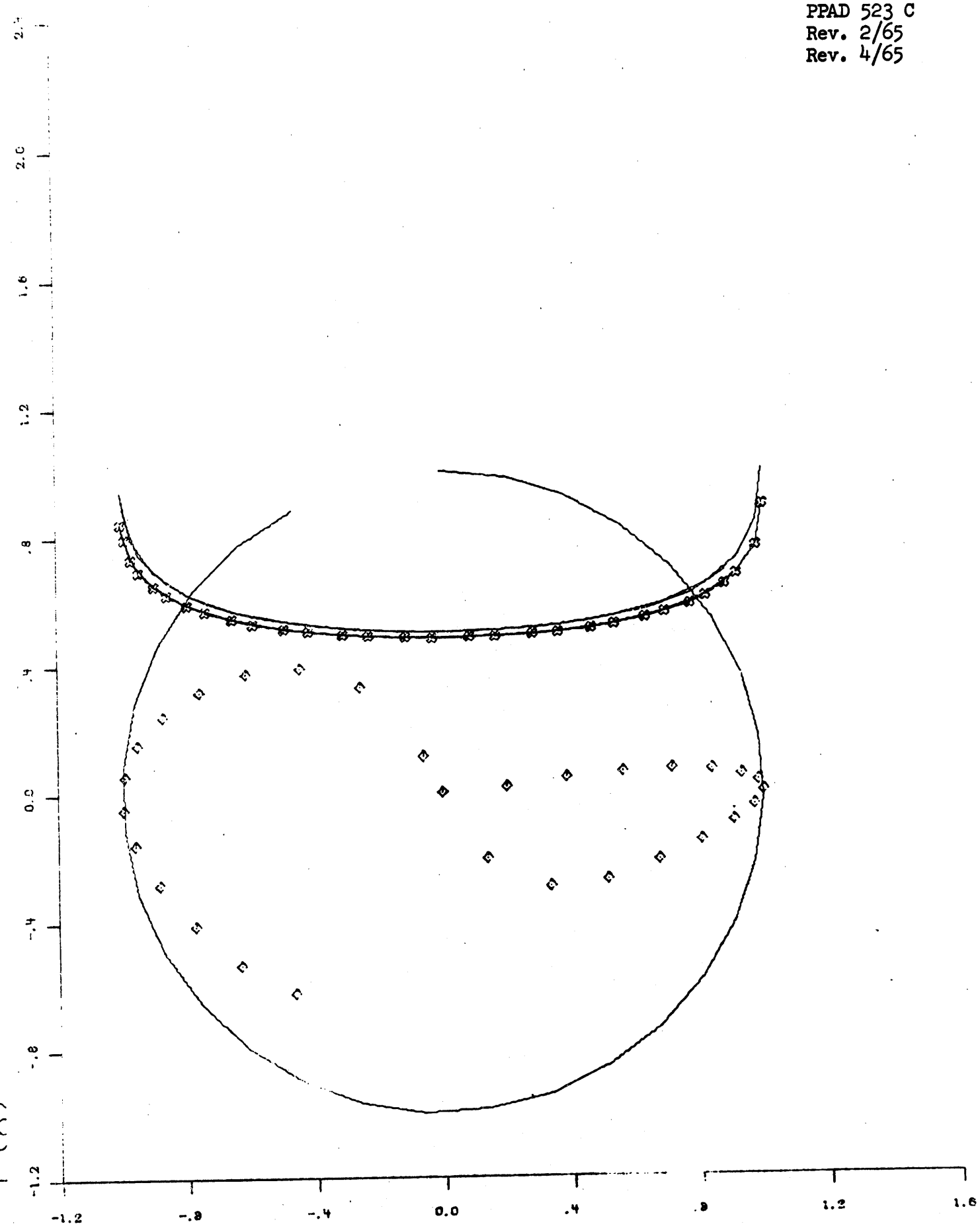
DRAWING NO. 2

PPAD 523 C
Rev. 2/65
Rev. 4/65



DRAWING NO. 3

PPAD 523 C
Rev. 2/65
Rev. 4/65



DRAWING NO. 4

Mrs. Joyce Fodor
Engineering Comp Lab
University of Wisconsin
Madison, Wisconsin

Planning and Running a Short-Term Workshop on Computer Orientation and Programming

Many groups find themselves wishing to conduct Short-Term Computer Institutes or Workshops of from 2 to 5 days, to satisfy the needs of a variety of different audiences. Typical situations are:

a) a College or University Computing Center running a special program for interested faculty,

b) an industrial program running an in-house training program for engineers and/or management,

c) sponsoring an advertised institute for outsiders, frequently with fees charged for attendance.

Based on the experience gained from several such undertakings, this paper will discuss the problems of determining the goal to be aimed at, the depth of penetration that can be reasonably achieved in a given number of days, how to plan the daily schedule, a typical graded set of problems, staffing considerations, and above all the importance of hands-on experience by the participants.

There are many questions to consider before planning a workshop. First one must consider what group the workshop is to be aimed at, and therefore what goals should be reached. If the group is composed of prospective programmers the institute should provide a start at programming competence. Supervisory personnel, on the other hand, though they may not become active programmers, need to achieve a sufficient knowledge of FORTRAN to understand a complex program written by a programmer, and also need to acquire an understanding of situations where computers can be profitably used. Management people need to know how and for what computers can be used in general company operations. The program should also be aimed at either scientific or business oriented people. The previous knowledge of the participants is an important consideration. If the group has little or no background (no actual programming experience) they will need a complete presentation of FORTRAN. If, however, they have written a few programs they need only a review of the language. Size is also an important consideration and is dependent upon the facilities available. With a 1620 Model I and a 1620 Model II with 1 diskdrive, a 407, and 8 keypunches we have found 25 is the maximum size that can be handled. The bottleneck is usually the keypunches.

The duration of the workshop is of course a balance of many factors: the amount of time the facilities and the instructional staff are available, the time the participants can afford to spend, the time needed to achieve the competence level established as the goal for the attending group.

For the purpose of this paper a specific group, size, duration, and situation will be assumed. Comments will be made where appropriate indicating how the planning discussed here should be modified if certain other factors existed. It will be assumed here that the group for which this workshop will be run consists of supervisory personnel with a scientific or engineering background. They are mainly from out of town, and have very little previous knowledge of FORTRAN. The facilities can be made available, and the group can make themselves available for three days.

Now that the group has been selected, the content of the workshop will need to be developed. Since the group knows little or no FORTRAN they will need to have lectures on the following topics:

1. Introduction, general concepts, intelligence of a computer, flow chart formulation of a problem,
2. Vocabulary of FORTRAN (excluding FORMAT and SUBROUTINE)
3. Examples of simple programs written in FORTRAN,
4. FORMAT,
5. Subroutines and Monitor operation.

Some lectures should also be devoted to theory. Since iteration underlies almost every useful computer application, a formal presentation of iterative techniques has been found very valuable. Its application in selected areas of mathematics and engineering would follow and should be connected to one or more problems. For the supervisory personnel a talk about some of the wider areas of applications and the future of computers is very desirable.

The most valuable contribution of a workshop of this type is the experience the participants get as they actually write and run their own programs. A group such as this can be expected to finish probably three problems during the three day period. The problems should be chosen so that the first is very simple and can be successfully run the first afternoon. Our experience has shown that this gives the participants a needed push onward. The next problems should be carefully selected so they increase in difficulty with no large jumps. One of the most common mistakes in planning an operation of this type is to make these problems too complex. The first problem in fact should be of such extreme simplicity, that there is no question what should be done, only how to say it to the computer. A set of problems suitable for this workshop is shown at the end. Each has several parts, with optional parts starred, to allow the participants to progress at individual speeds.

Developing the schedule is the next step to be taken. The first items to fill in are standard and include lunch at 12 each day and regular coffee breaks. Since this group is from out of town there should be a social hour and dinner, early in the session. This helps the group know each other and produces a closer knit group. If the participants were local and knew each other, at least a little, this social hour and dinner could well be later in the session so they have more knowledge and background and the guest speaker can pitch his remarks at a somewhat higher level.

Monday morning opens with registration, followed by an official welcome to get things started. The rest of the morning should be devoted to the first three topics in FORTRAN. This is in preparation for getting their first problem running the first afternoon. Due to the use of FORGO (a FORTRAN allowing free format and having very powerful diagnostic comments both at compile time and run time) the lecture on FORMAT can be postponed until a later day. The participants have enough to assimilate in the first day so if FORGO cannot be used it would be best to give the participants some standard FORMAT statements to use so the discussion of FORMAT can still be postponed. The schedule is shown allowing 1 1/2 hours for the introduction and vocabulary of FORTRAN before the break and about the same amount of time for examples and answering any questions.

The first afternoon the participants will need a demonstration of the use of the keypunches, lister and computers. If the facilities lend themselves to open-shop operation, there is much advantage to letting the participants do their own card handling and button pushing (under careful supervision) for at least the first problem, so that they get a clear picture of all the steps involved in the process. We prefer to maintain open-shop operation throughout the workshop, but some installations like to substitute in-out service later on. Careful attention to procedural details during this demonstration can anticipate and eliminate many questions that might arise when they start using the facilities.

The remainder of the afternoon is devoted to their solution of problem 1. Adjournment is listed at 4:30. We have found through experience, however, this usually is around 5:00 with some participants desiring to work even beyond this. The problem is to shut them off.

The second morning the participants need first a discussion of the solution of problem 1 with a period after for asking any questions that have arisen in their minds. They should then feel a little confidence in their mastery of the material already presented as they have successfully completed their first problem and had their questions answered. They should be ready to accept with understanding FORMAT, the single hardest statement in FORTRAN. This presentation needs close to 1 1/2 hours. After this a break is surely needed.

If the participants are to do the second problem the second afternoon a little background in theory and a presentation of the problem needs to follow the break.

The afternoon can then be devoted to a workshop for problem 2. It may be desirable to have a few keypunch operators available at this time to speed up punching of the complete programs for those participants who wish to use them. Several other keypunches should be available for participants who would rather punch their own and for all participants to make corrections and modifications.

The last half hour of the afternoon the group should reassemble for a presentation of problem 3. An optional workshop is scheduled for Tuesday evening. Many participants like to work in the evening to gain the maximum amount of knowledge during the three days. Presenting problem 3 at the end of the second afternoon and the optional workshop gives the participants three options: They can come to the evening workshop and complete problem 2 if they need more time for it. They can come to the evening workshop and try to write, punch, and run problem 3 if they wish to get a head start. If, however, they wish to stay in their room they can get program 3 written and it can be punched by keypunch operators while they attend lectures the next morning.

The first 45 minutes of the third morning is much the same as the second morning. They again need a discussion of preceding day's problem and an opportunity for questions and answers.

Since programmers and engineers are notorious for disliking to documentate their work, the supervisory personnel should be made aware of the value--indeed, the absolute necessity--of program documentation. This should take only 45 minutes. The participants now need time to finish their work on problem 3. This should only take 1 1/2 hours since they should all at least have their programs punched and some may have had several tries at debugging it. To finish the morning there is still one FORTRAN lecture remaining, to introduce them to subprogram structure and Monitor operation.

The last day's luncheon would profit from a guest speaker as the participants now have sufficient knowledge to understand a wider set of concepts. The last afternoon the participants will do problem 4, which is a modification of a preceding problem (number 3 in this case) to incorporate the use of a subroutine. This gives them a feeling for the use of subroutines in a minimum amount of time. After this workshop 30 minutes is needed for a discussion of problem 3, 30 minutes for summary and recapitulation, and 15 minutes for closing so this workshop session must end at 3:15 and the institute can finish at 4:30 to allow travel time home. Certificates should be presented at the closing because of the psychological effect on both the participants and the companies who spent the money to enable them to attend.

Now that the schedule has been completed the text and notes, staff and many other details must be decided on. Some of the texts we have used in the past are:

Davidson, Charles H., and Eldo C. Koenig, Computers, Wiley, N. Y., 1966.

Hamming, R. W., Numerical Method for Scientists and Engineers, McGraw-Hill, N. Y., 1962.

McCracken, D. D., A Guide to Fortran Programming, Wiley, N. Y., 1961.

McCracken, D. D., and W. S. Dorn, Numerical Methods and Fortran Programming, Wiley, N. Y., 1964.

Organick, E. I., A Fortran Primer, Addison-Wesley, Reading, Mass., 1963.

The lecturers and guest speakers must be chosen with care. They should know a great deal about the material they are presenting and still be able to present it at a beginner's level. During the workshop session there should be many laboratory assistants to help the participants with their problems. Care must be taken when these assistants are selected. If the assistants are too young and condescending in nature the participants will feel hesitant and embarrased about asking for help. The assistants must be mature acting and very willing to help. With a group of 25 participants we have found it desirable to have 5 assistants and 2-3 lecturers constantly available for questions and assistance. The powerful diagnostics that FORGO supplies at run time as well as compile time allows the participants to find many of their own errors. If it is not possible to use FORGO the number of lab assistants should be doubled. The time allowed for the completion of problems should also be increased.

Advance distribution of texts and notes to all registered participants is very desirable. It allows the participants to do some advance preparation, and makes clear the level of background and experience expected.

One thing we have learned from previous workshops is that it is important to keep the mathematics level from becoming too high so the participants do not have to worry about understanding the theory but can instead concentrate on FORTRAN. In a recent workshop we tried a problem in Differential Equations and found some of the participants were more confused by the mathematics than the programming.

The workshop that has been planned in this paper has been aimed at a particular group. If the workshop were being planned for business oriented personnel, an alternate list of possible problems is shown at the end. The topics of the theory lectures would also of course be changed. Since business oriented personnel are not as accustomed to analytical thinking as engineers, the problems should progress at a slower pace. Simulation of a traffic problem is a very good topic for a guest speaker and demonstration on the machine but a little deep for programming.

If the workshop is planned as an in-house training program it is many times desirable to use only part of each day as the lecture or workshop session as more time is available between sessions for additional thinking and study.

Actually I have said nothing about running the workshop. If it is carefully planned the workshop will run itself.

SCHEDULE

3 Day FORTRAN Workshop

MONDAY Morning

8:00 Registration
8:30 Welcome
8:45 FORTRAN
10:15 Break
10:35 FORTRAN
12:00 Lunch

MONDAY Afternoon

1:30 Demonstration of use of the facilities
2:00 Workshop Problem 1
4:30 Adjourn

MONDAY Evening

6:00 Social Hour
7:00 Dinner
--Guest Speaker--

TUESDAY Morning

8:00 Discussion of Problem
8:25 Questions and Answers
8:45 Further FORTRAN: FORMAT
10:10 Break
10:30 Iterative Techniques
11:30 Problem 2 Presentation
12:00 Lunch

TUESDAY Afternoon

1:30 Workshop Problem 2
4:00 Problem 3 Presentation
4:30 Adjourn

TUESDAY Evening (optional)

7:00 Workshop finish Problem 2 or start Problem 3

WEDNESDAY Morning

8:00 Discussion of Problem 2
8:25 Questions and Answers
8:45 Value of Documentation
9:30 Workshop Problem 3
11:00 Further FORTRAN: SUBROUTINES
12:00 Lunch
--Guest Speaker--

WEDNESDAY Afternoon

1:45 Workshop Problem 4 a preceding problem modified to use subroutines
3:15 Discussion of Problem 3
3:45 Summary & Recapitulation
4:15 Presentation of certificates

NOTE: Coffee served during workshop sessions

A TEN YEAR BUDGET PROJECTION

(A Role for the Computer in Long-Range Planning for Educational Institutions)

John A. Ferling

Preface

Since its founding in 1947, Claremont Men's College, an independent four-year liberal arts college, has prepared several long-range plans for its future. These plans, as well as recent applications for grants to the Ford Foundation, have included ten-year budget projections. The computations necessary for the budget projections were carried out on a desk calculator.

The author of this paper has prepared a program for the IBM 1620 which carries out these computations in minutes rather than days. This program has enabled the college to study the budgetary effects of changes in items ranging from the expected inflation to percentage of tuition incomes for scholarships. With the help of the computer Claremont Men's College has explored the feasibility of several plans. Special consideration was given to the question of optimal size. The program discussed in this paper has been written for a specific college and hence changes will be necessary before it can be used by another institution. S. Tickton[1] has discussed projections of this kind in great detail without using computers.

Machine Requirements

The program is written in UTO Fortran for a 20K 1620 and uses card input and typewriter output. Special features are not required.

Information Required and Discussion of Program

The information to be supplied by the administration includes:

A. Per year for the next ten years

1. Number of students
2. Tuition charge
3. Allocations for scholarships as percentage of tuition income
4. Student-Faculty ratio
5. Percentage increase in administrative salaries
6. Percentage increase in average faculty salary
7. Total percentage increase in development office salaries
8. Total percentage increase plant maintenance salaries
9. Total percentage increase in business office and student welfare expenditures
10. Total percentage increase in library expenditures
11. Expected inflation
12. Expected gifts to college
13. Rate of return on endowment

[1] Sidney G. Tickton, Long-Range Planning: A Case Study, pp.138-161, in Financing Higher Education 1960-70, McGraw Hill, New York, 1959

B. For first or present year

Salary total and non-salary total for

14. Administration
15. Admission
16. Registrar
17. Instructional (excl. faculty salaries)
18. Dean of Students
19. Development
20. Plant Maintenance

and

21. Average Faculty Salary
22. Physical Education non-salaries
23. Miscellaneous
24. Certain Fringe Benefits (called "Insurance Fudge" in program)
25. Library (Total)
26. Business Office (Total)
27. Student Welfare (Total)
28. Initial Endowment

In addition to this information, estimates of the marginal cost per additional student for certain salary and non-salary expenditures are needed. These estimates are denoted by WGS (weight on growth salary) and WGNS (weight on growth non-salary), respectively, and are expressed in percent. The weight on growth for salary items in the sample projection given below is 50 percent. The salary total for the admissions office for a student body of 1000 is $30,000.00, i.e.,$30.00 per student. The addition of 50 students for the second year would then increase the salary total by $750.00 (50% of 50 times $30.00). However, this amount is not allocated to the admissions office since one would not employ a fraction of an admissions officer. These $750.00 are allocated to a fund called "General Administrative Growth Allowance". The salary total of the admissions office is only increased by the percentage increase in administrative salaries which is assumed to be 5% in this projection. This yields the $31,500.00 entry for the second year. Several other salary items (Registrar, Instr. Non-Fac., Stud. Dean) are treated in the same fashion. The resulting amount in "General Administrative Growth Allowance" is also increased by 5 percent (percentage increase in administrative salaries) and is equal to $4,077.22 in the second year. The item "Administration Salaries" (president and assistants) is assumed not to be effected by a larger student body, whereas "Administration Non-Salaries" is. The weight on growth, WGNS, for several non-salary items in this sample is 75 percent. Many of these are also subject to the 1 percent inflation

increase. Most of these computations are carried out by means of a "Procedure" containing a multiplier which makes it possible to increase, decrease or remove the effect of the growth factors and inflation to suit the assumptions of the planner. A study of the source program, which will be supplied by the author upon request, will indicate which changes effect the individual entries.

Auxiliary enterprises such as dining hall, dormitories, and bookstore are assumed to be essentially self-balancing items. A more intensive use of existing classroom facilities will make the addition of extra buildings not absolutely necessary. However, since it is relatively easy to find a donor for a building which would be named after him, it is expected that a classroom building will eventually be added.

The program computes per year for the next ten (or nine in certain cases) years the following:
Tuition income, scholarship allocations and number of faculty; salary and non-salary expenditures for administration, admission, registrar, faculty, dean of students, development and plant maintenance; expenditures for physical education, business office, library, student welfare, fringe benefits and miscellaneous; gifts to current operations used to balance the budget and endowment at end of year.

Since the output is under control of a sense switch, the effects of various changes in input data can be studied without having to wait for a complete typeout. A complete output is given below. The data used are fictitious.

TEN-YEAR BUDGET PROJECTION FOR A COLLEGE
INPUT AND OUTPUT
(INPUT DATA ARE UNDERLINED AND APPEAR AGAIN IN OUTPUT)
(FICTITIOUS DATA)

WGS	WGNS			
50.00	75.00			
NO. OF STUDENTS				
1000.	1050.	1075.	1100.	1125.
1150.	1150.	1200.	1200.	1250.
TUITION				
1500.	1550.	1600.	1650.	1700.
1725.	1750.	1775.	1800.	1800.
TUITION INCOME				
1500000.	1627500.	1720000.	1815000.	1912500.
1983750.	2012500.	2130000.	2160000.	2250000.
P.C.FOR SCHSH.				
13.00	14.00	14.00	15.00	15.50
16.00	16.50	17.00	18.00	18.50
SCHOLARSHIPS				
195000.00	227850.00	240800.00	272250.00	296437.50
317400.00	332062.50	362100.00	388800.00	416250.00
STUD.-FAC. RATIO				
12.00	12.20	12.40	12.60	12.60
12.90	12.90	13.50	13.50	13.50
NO. OF FACULTY				
83.33	86.06	86.69	87.30	89.28
89.14	89.14	88.88	88.88	92.59
	SALARIES			
P.C. INCR., ADM.				
.00	5.00	5.00	5.00	4.00
4.00	4.00	5.00	5.00	5.00
ADMINISTRATION				
62000.00	65100.00	68355.00	71772.75	74643.66
77629.40	80734.58	84771.31	89009.87	93460.36
ADMISSION				
30000.00	31500.00	33075.00	34728.75	36117.90
37562.61	39065.12	41018.37	43069.29	45222.75
REGISTRAR				
17000.00	17850.00	18742.50	19679.62	20466.81
21285.48	22136.90	23243.74	24405.93	25626.22
INSTR. NON-FAC.				
25200.00	26460.00	27783.00	29172.15	30339.03
31552.59	32814.70	34455.43	36178.20	37987.11
STUD. DEAN				
21123.00	22179.15	23288.10	24452.51	25430.61
26447.83	27505.74	28881.03	30325.08	31841.34

ADM.SAL.GR.ALLOW.				
.00	4077.22	6421.63	8990.28	11687.37
14585.84	15169.27	21236.98	22298.83	29267.22
P.C. INCR.,DEV.				
.00	20.00	5.00	5.00	5.00
4.00	4.00	4.00	4.00	5.00
DEVELOPMENT				
82235.00	98682.00	103616.10	108796.90	114236.74
118806.20	123558.44	128500.77	133640.80	140322.84
P.C.INCR.P.M.				
.00	6.00	11.00	4.00	4.00
4.00	5.00	5.00	5.00	5.00
PLANT MAINTENANCE				
63500.00	67310.00	74714.10	77702.66	80810.77
84043.20	88245.36	92657.62	97290.50	102155.03
P.C.INCR.FAC.				
.00	7.00	7.00	7.00	7.00
5.00	5.00	5.00	5.00	6.00
FACULTY-AVERAGE				
12500.00	13375.00	14311.25	15313.03	16384.94
17204.19	18064.40	18967.62	19916.00	21110.96
FACULTY-TOTAL				
1041666.	1151127.	1240693.	1336852.	1462941.
1533707.	1610392.	1686011.	1770311.	1954719.
	NON-SALARIES			
P.C. INFLATION				
.00	1.00	1.30	1.30	1.30
1.00	1.00	1.00	1.50	1.50
ADMINISTRATION				
10000.00	10100.00	10231.30	10364.30	10499.04
10604.03	10710.07	10817.17	10979.42	11144.11
ADMISSION				
11000.00	11526.62	11895.40	12275.98	12668.73
13035.33	13165.68	13795.99	14002.93	14745.96
REGISTRAR				
2145.00	2247.69	2319.60	2393.81	2470.40
2541.89	2567.30	2690.21	2730.57	2875.46
INSTR. NON-FAC.				
46123.00	48331.13	49877.43	51473.19	53120.01
54657.17	55203.74	57846.62	58714.32	61829.85
STUD. DEAN				
3245.00	3400.35	3509.14	3621.41	3737.27
3845.42	3883.87	4069.81	4130.86	4350.06
P.E.				
31123.00	32613.01	33656.42	34733.21	35844.46
36881.71	37250.52	39033.89	39619.40	41721.70
MISC.				
43123.00	43554.23	44120.43	44693.99	45275.02
45727.77	46185.04	46646.89	47346.60	48056.79

PLANT MAINTENANCE				
49500.00	49995.00	50644.93	51303.31	51970.26
52489.96	53014.86	53545.01	54348.18	55163.40
DEVELOPMENT				
43000.00	43430.00	43994.59	44566.51	45145.88
45597.34	46053.31	46513.84	47211.55	47919.72
INSURANCE FUDGE				
16666.66	17385.24	17739.75	18096.40	18748.28
18906.41	19095.47	19230.52	19518.98	20637.26
P.C. INCR.B.O.-ST.WELF.				
.00	4.00	4.00	4.20	4.50
4.80	4.80	4.70	4.00	4.00
BUS. OFFICE				
52000.00	54080.00	56243.20	58605.41	61242.65
64182.30	67263.05	70424.41	73241.39	76171.04
STUDENT WELFARE				
60258.00	62668.32	65175.05	67912.40	70968.46
74374.94	77944.94	81608.35	84872.69	88267.59
P.C. INCR.,LIBRARY				
.00	7.00	7.00	7.00	7.00
7.00	7.00	7.00	7.00	7.00
LIBRARY				
86000.00	92020.00	98461.40	105353.69	112728.44
120619.43	129062.79	138097.18	147763.98	158107.45
	ENDOWMENT			
RATE OF RETURN				
5.00	5.00	5.00	5.20	5.20
5.00	5.00	5.00	5.00	5.00
GIFTS				
900000.00	950000.00	1000000.00	1100000.00	1200000.00
1400000.00	1500000.00	1500000.00	1600000.00	1800000.00
INIT. ENDOWMENT				
8000000.				
ENDOW.INC.				
422500.	465279.	509494.	579489.	634333.
668401.	733184.	798814.	868395.	942824.
GIFTS TO CUR. OP.				
69408.26	90707.30	95862.79	95302.19	130697.38
154332.71	187401.04	158381.32	211415.20	315017.10
ENDOWMENT AT END OF YEAR				
8830591.	9689884.	10594021.	11598719.	12668022.
13913690.	15226289.	16567908.	17956493.	19441476.

JUNIOR COLLEGE INSTRUCTIONAL TESTING PROGRAM

Prepared by the Computer Center Staff
Miami-Dade Junior College

With computers installed or on order for the vast majority of colleges and universities, educators must decide on the usage of this powerful instructional and administrative aid. The cost of a computer installation represents a large item in the institution's budget, and in most junior colleges sufficient funds are not available for two computer installations; one for administrative work, and one to support the instructional program.

By developing a computer concept of providing a computer center with the responsibility of processing administrative work and scheduling classes for instruction, a more extensive and modern computer can be made available. The Miami-Dade Junior College has developed this centralized computer concept and the results have been most encouraging.

As is indicated on the organizational chart, all instruction in the center is the responsibility of the coordinator of computer science, which includes a business and scientific option two-year occupational program. Administrative service is the responsibility of the manager of data processing. Some seventy-four administrative programs provide a wide variety of services for the college, and one that has made the greatest impact on the faculty as a whole has been the program of test writing and test scoring.

The responsibility for working with the various departments within the college is assigned to the systems analyst. He works with the personnel within the departments to establish procedures that may be handled in the computer center. After the precise input and output have been determined, flowcharts are developed and the writing of the programs is assigned to the programmers.

Miami-Dade Junior College began test scoring on the IBM 1620 in the Winter Term of 1962. Professor Joseph Duerstock undertook a rewriting of the program to overcome several operational difficulties and has developed five versions of the test scoring program now in use. Our college enrolled over 13,000 students this past fall, and over 40,000 tests were scored and analyzed using our system.

An automated test scoring system provides a partial solution to the ever-increasing demand on instructional staff by transferring test scoring and test analysis to machines.

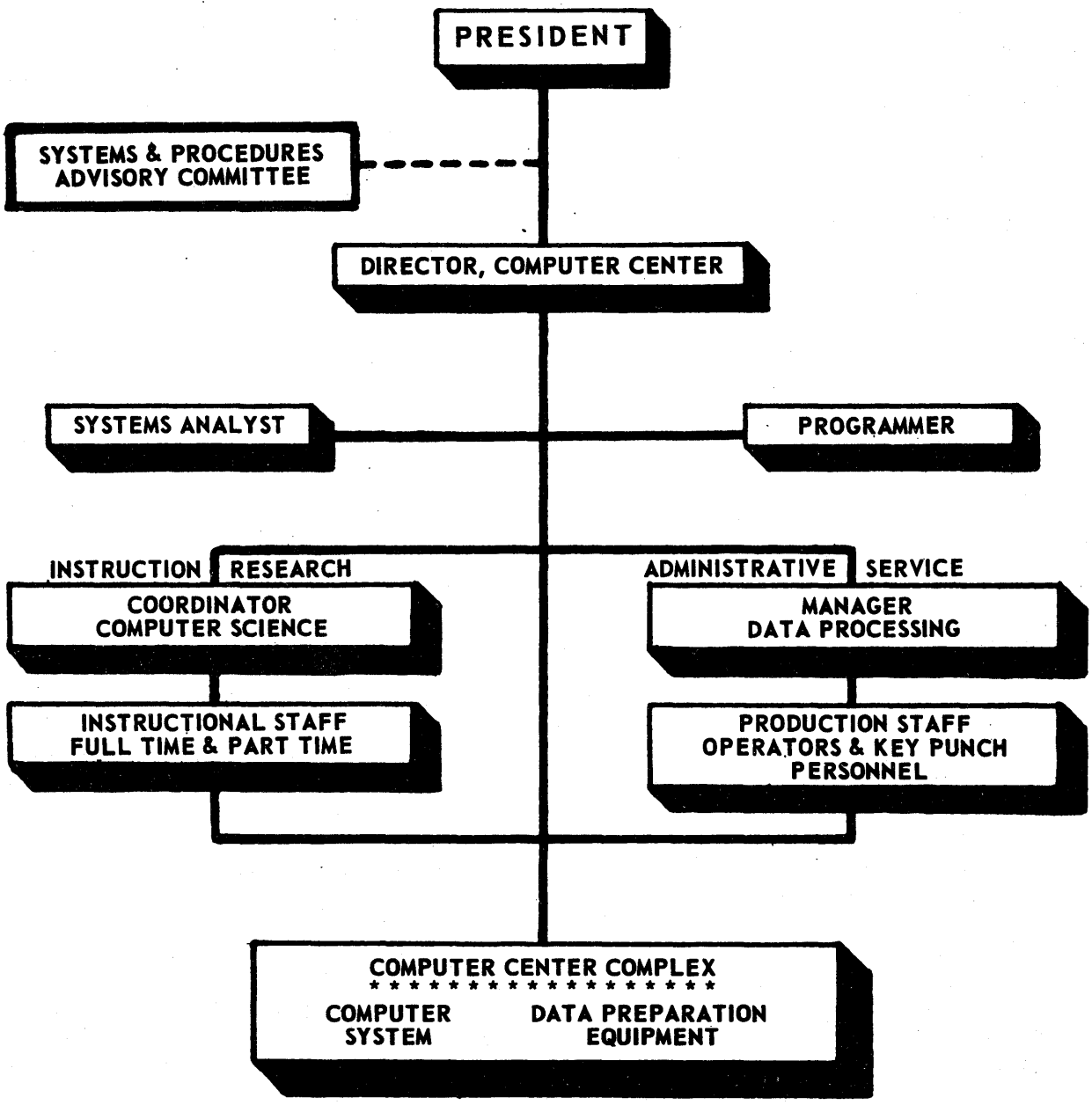
The application of modern computer systems to test scoring provides a rapid method of grading large numbers of tests while accumulating statistical data that heretofore could only be gathered through tedious error-prone manual calculations. Not only will computers provide this information as part of the normal processing, but mathematical formulas which were infrequently used because of their complexity now provide valuable data for use in the analysis of tests. Thus, it is possible to reduce the work load of the faculty and, at the same time, produce data that can be used to improve the tests.

Miami-Dade Junior College's test scoring system had been designed to accommodate true-false or multiple choice answer tests. The main restriction in the construction of the test is that there can be only one correct answer to each test question. Thus, most objective test questions can be readily adapted to the system while those with more than one answer can be rewritten.

MIAMI-DADE JUNIOR COLLEGE

11380 N.W. 27th AVENUE • MIAMI, FLORIDA 33167 • PHONE 688-3541

ORGANIZATIONAL CHART



INSTRUCTION

HUMANITIES
NATURAL SCIENCE
SOCIAL SCIENCE
TECHNICAL
BUSINESS
SPECIAL PROJECTS
COMMUNITY SERVICES
TESTING

RESEARCH

ADMINISTRATIVE
STUDENT PERSONNEL
GUIDANCE
TESTING
REGISTRATION
SPECIAL PROJECTS

ADMINISTRATIVE SERVICE

REGISTRATION
FINANCE
CAMPUS SERVICES
PLANNING
PAYROLL & PERSONNEL
INVENTORY
ALUMNI
MAINTENANCE

TAK:DS

July 1, 1964

(See sample test question formats.)

Equipment requirements necessary to implement this test scoring system are as follows:

1. IBM 1620 Computer Complex
2. IBM 407 Accounting Machine or IBM 1443 Printer
3. IBM 519 Document Originating Machine
4. IBM 026 Card Punch
5. IBM 082 Sorter

Because of the different machine capacities and features of the 1620 complex, several versions of the test scoring system were developed. The test scoring system is comprised of two totally independent, wholly compatible, units which are called (1) the test writing program and (2) the test scoring program.

The former program was designed to use the data processing equipment to relieve the typing work of the secretary. The benefits of this procedure became evident during the ever-occurring semester-end rush to prepare the final examinations prior to the deadline of the duplicating department. Once the questions to be used were determined, the entire job of machine processing and preparation of the mats for duplicating was completed easily within thirty minutes. This processing often included tests that contained several hundred questions--a difficult task for any secretary or secretary pool to perform on short notice.

An additional advantage of the system is accuracy. Once correctly typed, the test questions are duplicated with no typing errors and the computer plans all spacing to insure that no question would be split across pages. The computer also causes every sheet to be identified by a heading and a page number.

The test questions are converted into the proper card format for processing, using an 026 card punch. This file is then reprocessed on the IBM 1620 computer to create a master processing file that contains all questions, each one of which has a unique question number. The instructor selects the desired questions by number and records this information on a sheet of paper. The operator enters the numbers into the computer, and the computer compares each card to find the corresponding questions. The data from the processing file is transferred in slightly altered form with all of the identification deleted to another card. These output cards are consecutively numbered to facilitate sorting in the event the cards get out of order.

The final step is the transfer of the test from the output cards to some form for duplicating. The IBM 407 accounting machine or the IBM 1443 printer is used to produce the final printed test on continuous form offset duplicating mats. Ditto mats may also be used if fewer copies are desired. If necessary the regular single form mimeograph masters may be used in the same manner as they are used on a typewriter. The ribbon must be removed from the machine.

THE TEST SCORING PROGRAM

This portion of the system provides the method through which the tests are graded and the analysis generated. Since one of the primary reasons for the use of computers for scoring is speed, considerable programming effort was devoted to reducing the processing time per test card. The following

scoring speeds were obtained using version five of the test scoring system, with an IBM 1443 printer for direct output.

1 card test	--	50 questions	--	236 students per minute
2 card test	--	100 questions	--	128 students per minute
3 card test	--	150 questions	--	88 students per minute
4 card test	--	200 questions	--	62 students per minute
5 card test	--	250 questions	--	54 students per minute

Perhaps the most interesting thing that this chart shows is the relationship between the maximum speed of the computer input (250 cards per minute) and the output (240 lines per minute) to the number of cards per student. It can be seen that the process is output bound for one card test and input bound for all other cases. Therefore, additional improvement in processing speed would require faster input/output devices rather than improved programming.

The most critical part of the test scoring program is the marking of cards by students; however, experience has shown that the students readily adapt the required techniques when properly instructed.

Though the principle of the mark sense card is the same as that of the well-known 8½ by 11 inch scoring sheet, there are two significant differences:

1. A student cannot mark two responses to the same question without discovery. This is handled by either of two methods, depending upon the equipment available for use:
   a. If a fully equipped 519 document originating machine is used, only one of the marks is accepted for scoring and the second causes the punching of a digit in card column 9 of the answer card. This digit is later used by the computer to generate an error message associated with the student's grade.
   b. If the 519 does not have this feature, both marks will be punched into the card. Double punching may form an invalid character which will cause the computer to stop during the processing of that card. Though this process will catch the error, it also causes the computer to become inoperable to the extent that the program must be reloaded and processing restarted from the beginning. This is the most convincing reason for using the full capacity 519.
2. The cards must be carefully handled so that they are not bent or frayed when processed.

MARK SENSE PUNCHING

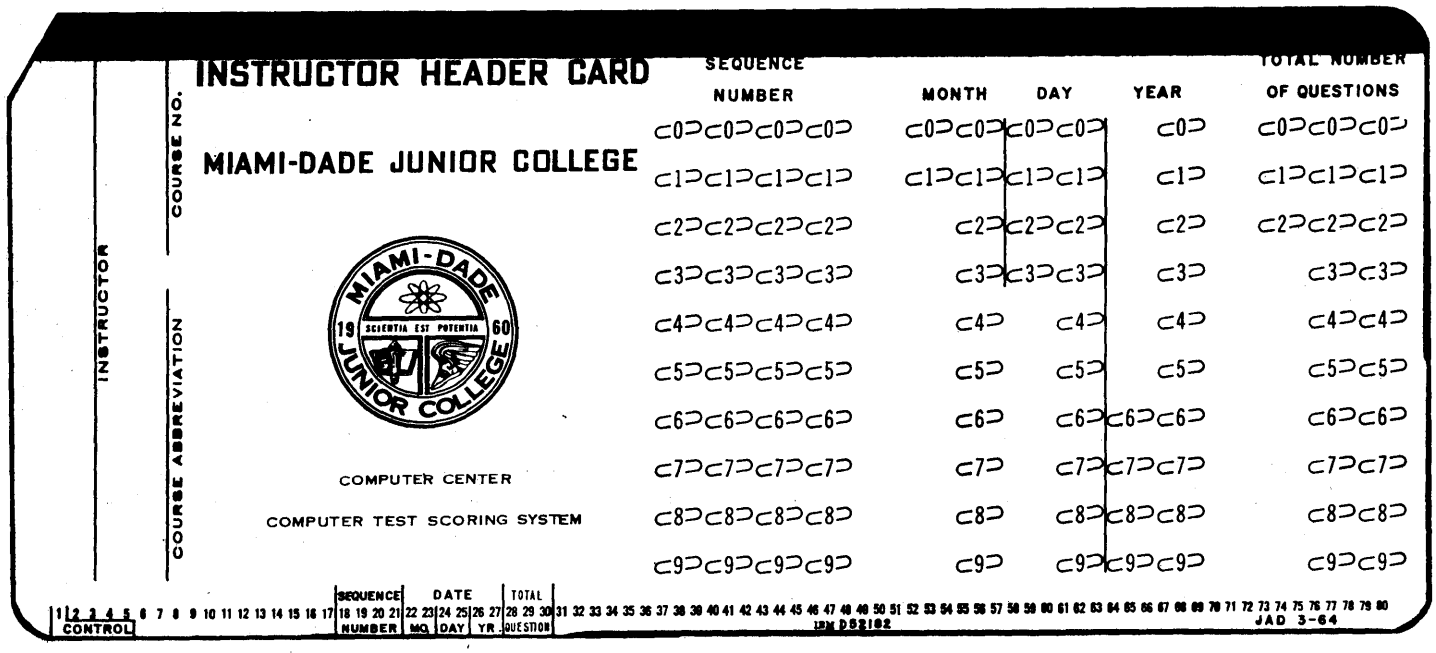
Mark sense punching, though not a new innovation, provides a highly accurate method of recording and scoring student answers and, at the same time, provides two records of the student's response--the original pencil mark and the punched hole generated from that mark. This system insures that no student can change an answer (the mark) on the card and confront the instructor with an error in his score because the computer failed to function properly. The hole cannot be changed by the student.

The reliability of mark sense punching is excellent if the faculty and students are properly introduced to the correct manner of marking the cards. Mark sensing is being used successfully in colleges, high schools, junior high schools, and even in the lower grades. Favorable results can be achieved

by insuring that:

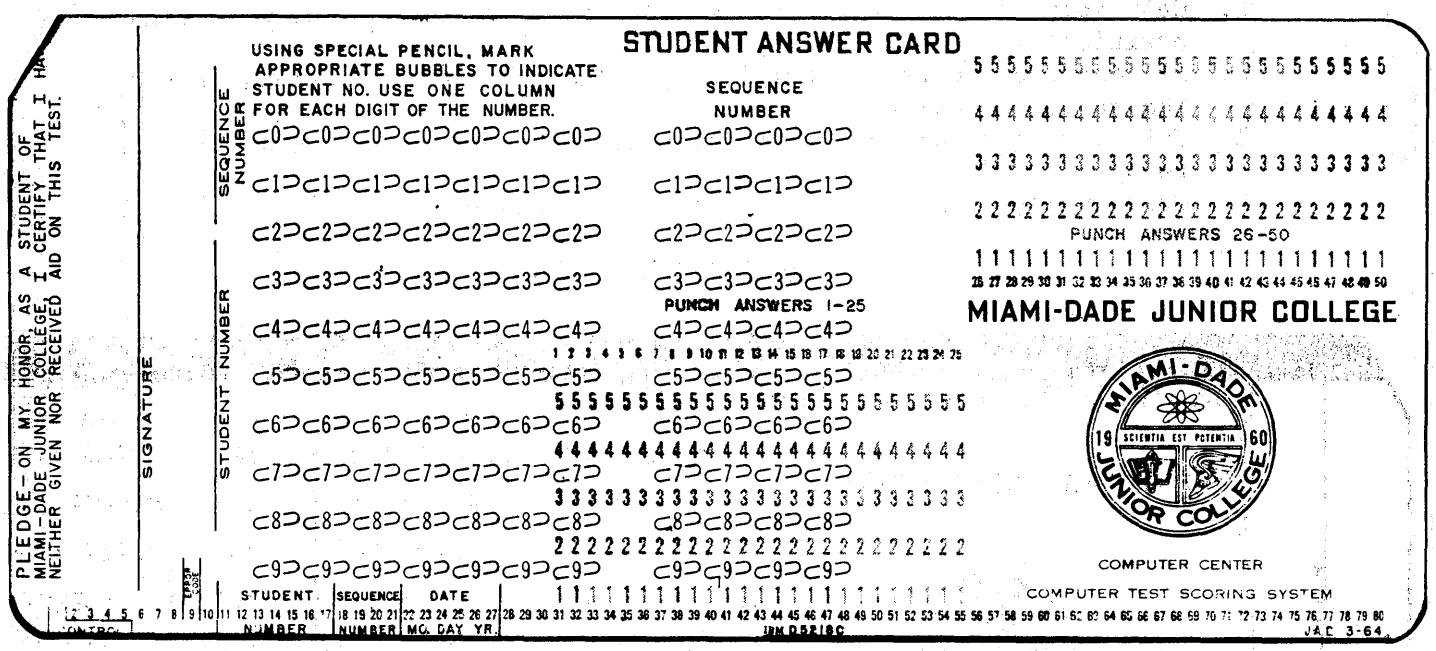
1. Only special Electrographic pencils are used to mark the mark sense positions.
2. All marks are dark and completely filled, but do not extend beyond, the position outline.
3. Ink is used to write any other information on the card that is required.
4. Only one response is marked for any one question.

INSTRUCTOR HEADER CARD



The instructor fills in one of these cards for each class tested. The sequence number, date, and the total number of questions are recorded by the instructor in the appropriate mark sense positions. The total number of questions is the total of those questions to be scored, not the numerical value of the highest question number. It is used to check that the marks on the key card(s) punch properly. For example: If a thirty question test were given and it was desired to delete question number ten, the instructor would simply not give number ten an answer on the key card and then enter the number twenty-nine on the instructor header card for the total number of questions. This feature does not insure that the proper responses have been marked, but rather that there is a mark for each question to be scored. Having to reprocess a test because the instructor failed to use the Electrographic pencil properly is prevented by this checking procedure. The use of the instructor header card is not mandatory, but if used, total number of questions must be entered in first instructor header card following the key card(s).

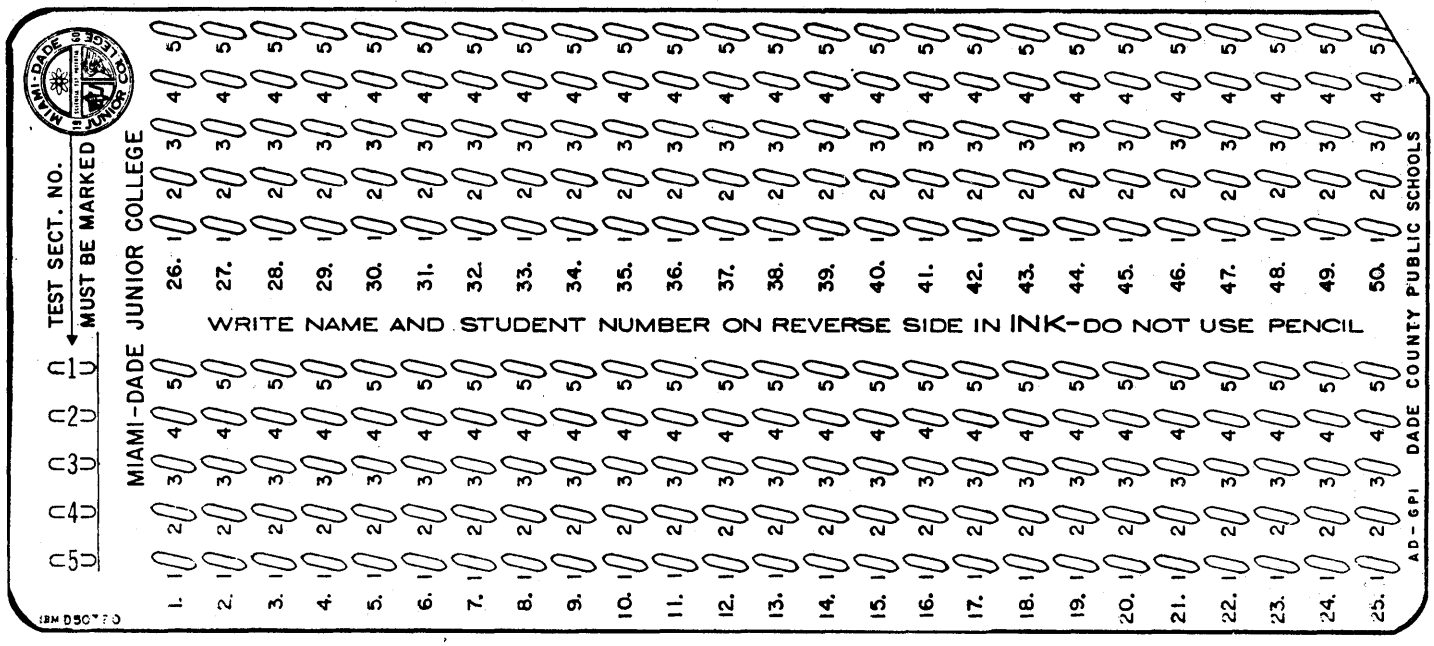
STUDENT ANSWER CARD
(Front Side)



The student enters his last name, first name, middle initial, student number, and sequence number on the appropriate lines preferably using a pen. The special pencil could be used if the student is careful not to allow a stray mark to cross into the area of mark sense positions where the student number is recorded.

Using the special pencil, the student then marks his student number and class sequence number into the designated area making one mark in each of the appropriate vertical positions. All zeros, including leading ones, must be entered. The program checks each position of the student number for a mark and, if any of the seven positions are blank, an invalid student number is generated and the processing of the card is terminated. Consequently, the student will have missing answer cards if the test uses more than one answer card. This procedure is necessary to prevent the accumulation of an incorrect total score if two consecutive students completely omit their student numbers.

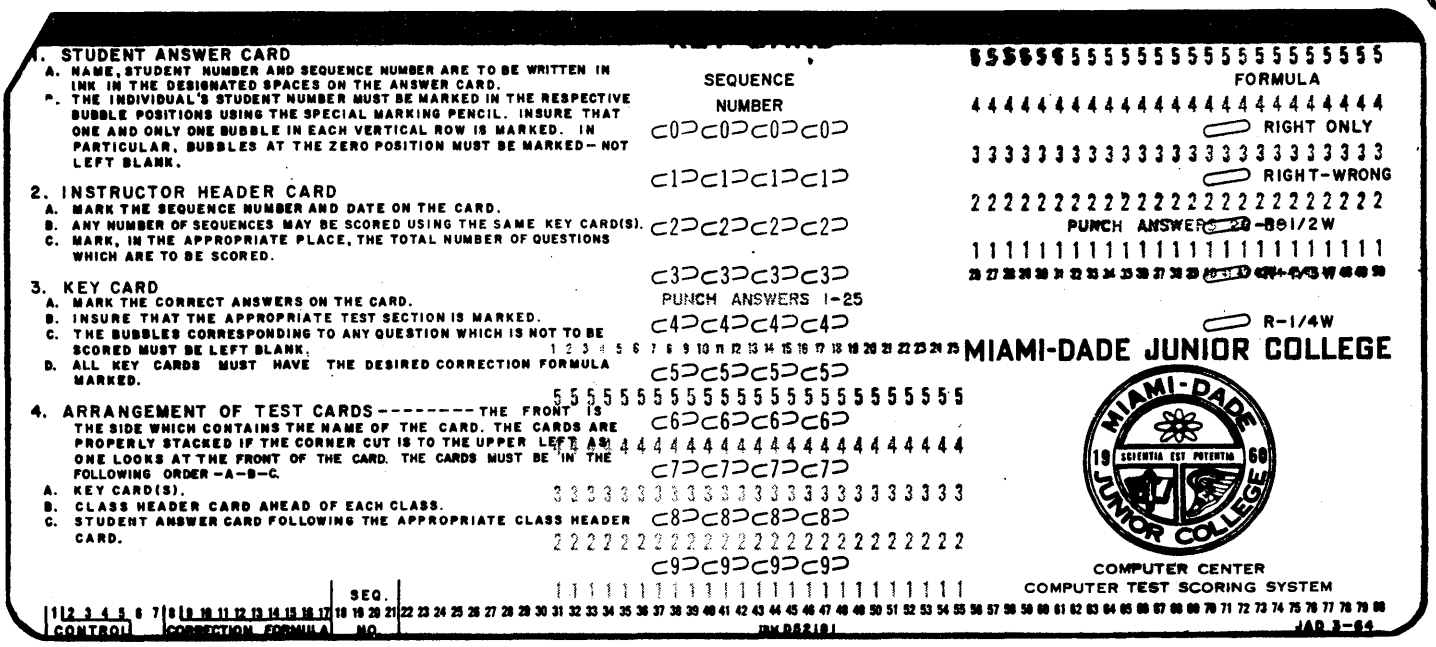
KEY CARD AND STUDENT ANSWER CARD
(Reverse Side)



The responses are entered for the appropriate questions and a test section number must be marked. It is imperative that the students use the same test section that is marked by the instructor, though it is not necessary to start with test section number one.

The use of the test section number provides an instructor with the capability of giving a 250 question examination. A practical and useful variation is to give not more than five 50-question tests. By marking a different test section number for each test, the cards from all exams can be rerun at the end of the semester to provide cumulative scoring to facilitate the assigning of the final grade.

KEY CARD
(Front Side)



The instructor marks the desired correction formula on the key card by filling in the appropriate mark sense position. If no foumula is marked, the computer assumes "right only." If the instructor wishes to use one of the remaining formulas, all of the key cards should be marked as the computer processes this mark only from the first key card entered for the test group. All figures are calculated to give the benefit of the doubt to the student, that is, percentages are half-adjusted upward; or, when a correction formula is used, the fractional portion of the subtrahends used to correct for guessing are dropped.

OUTPUT LISTING

Page one of the output listing shows one line of print for each student. The student number, number right, number wrong, and the number of questions with no answer are listed. The last column listed is the percentage and is calculated by dividing the number right by the total number of questions. If a correction formula is used, the percentage column is changed to reflect the adjusted score. Statistics are based on the number right if no correction formula is specified, or the adjusted number right when a correction formula is used.

ERROR STATEMENTS

ER 4 - Student is missing one or more answer cards. All statistics for the student are ignored and no grade is assigned.

ER 3 - The student has marked more than one answer on one or more

questions. The instructor should examine the student's card to try to determine if the double marks were intentional or a legitimate mistake.

ER 2 - The student has two or more answer cards with the same test section number. ER 4 will normally be listed along with this error.

Student Number Invalid - This statement appears with an asterisk to the left of the student number if the student fails to fill in one or more of the student number mark sense columns. The computer will automatically substitute a zero for any digit missing from the student number.

Student Number XXXXXXX has improperly marked test section number on answer card - The student number of the person who made the error is inserted in the X'd space by the program. It is possible for the statement to be blank or to appear with a partially filled number if the student number is also improperly marked. The card is not scored.

The item analysis provides a tabulation for the responses to each item of the test. An asterisk marks the correct response and is determined by the answers provided by the key card(s). The mean, median, standard deviation, sum, and sum of the squares, as well as the number of test questions and tests graded, are given at the bottom of the last item analysis sheet. Individual scores which were deleted because of errors are not included in the statistical analysis.

The last column (R) is a point biserial correlation between the response to each item and the score made on the test. It is not the purpose of this paper to debate the significance of R for various testing situations; however, a number of interesting properties of R are immediately evident.

If a significantly high value of R is found for an item, it follows that the students who answered the item correctly had the better scores on the test. However, this is no assurance that the question measured that which was intended.

A zero value from one of the following conditions:

1. All students answered the item correctly.
2. All students missed the item completely.
3. The correlation was equal to zero or so near zero that the only significant digits were values less than three decimal places.

The distribution is presented in two forms. A numerical listing is given showing the frequency for each score, omitting all scores with a zero frequency value. The last column contains the cumulative frequency for the convenience of those who assign grades based upon class standing.

A histogram is generated to provide a graphic illustration of the distribution. The horizontal axis represents the score and the vertical axis the number of students. The computer adjusts the scale of either axes in the event that more than 50 questions are used, or in the event there are more than 50 people that score a particular grade. Thus the histogram, like the other portions of the output, always fits an 8½ x 11 inch page.

AVAILABLE TEST SCORING PROGRAM VERSIONS

Version I - For 40K machine. Has item analysis, mean, median, standard deviation, total number of questions and tests graded. Distribution has both frequency and

cumulative (equal to or greater than) frequency distribution by score and a biserial coefficient of correlation is calculated for each item of the analysis. Program can be reinitialized by recalling from the 1311 disk drives or by reloading from cards following the grading of each test group.

Version II - Similar to Version I except a histogram plot is generated for visually representing the distribution. Part of the program is overlaid automatically from disk or manually from cards. The program is in two parts--scoring and analysis.

Version III - For 60K machine without disk drive. Similar to Version II except total program is contained in core.

Version IV - For 20K machine with disk drive. Program is limited to three cards (150 questions) and scoring speed is somewhat reduced for tests that have more than one card. Program is overlaid from disk to produce the same analysis as Version II.

Version V - For use with the 1443 printer to permit direct printing of output, and the 1311 disk drives to permit automatic overlay of the analysis routine and reinitialization. In other respects, the system is similar to Version II.

DETAILED MACHINE FEATURES REQUIRED FOR TEST SCORING PROGRAM

1620 Central Processing Unit with following special features:
1. Automatic divide
2. Transmit numeric fill (TNF)
3. Transmit numeric strip (TNS)
4. Indirect addressing
5. Move flag (MF)

1622 Card Read-Punch (either Model 1 or Model 2)

519 Document Originating Machine with:
1. Mark sensing -- 27 positions
2. Double punch-blank column detection device
3. The following features provide additional checking capabilities and ease of operation. They are strongly recommended, but are not absolutely required. If only one of the two can be secured, the selectors are preferable:
   a. Four selectors -- allow the panel to be wired to eliminate the possibility of punching a double-marked response.
   b. Punch emitter -- provides ease in operating the machine and in wiring the control panel for the punching of the mark sense cards.

1443 Printer or 407 Accounting Machine -- If the accounting machine is used, only an 80-80 list board with a carriage skip to channel 1 from column 80 first read is needed.

026 Card Punch -- No special features required.

082 Sorter -- No special features required.

ACCEPTABLE TEST QUESTION FORMS

1. Multiple Choice:
   How many characters comprise the alphabet.
   1. 9
   2. 10
   - 3. 26
   4. 39
   5. 47
2. True-False
   Florida is known as the Sunshine State.
   - 1. True
   2. False
3. Multiple-Multiple Choice
   Testing procedures should include
   A. Test writing
   B. Grading
   C. Evaluating
   D. Giving A Grades to all students
      1. A and D
      2. B, C, and D
      3. B only
   - 4. A, B, and C
      5. All of the above

TEST WRITING PROGRAM STEPS

This is a detailed description of the steps used in the test writing system, starting with the questions already written.

1. Each test question is transcribed to IBM cards using the following card format.
   1-10 blank in all cards
   Format for question cards:
   11-80 contain the question. If it is necessary to have two or more lines for the question, the succeeding lines also start in column 11.
   Format for response cards:
   16-17 contain the response number and a period.
   20-80 contain the response. Additional lines for responses start in column 20.
   An 11 (-) punch is placed in column 11 of the card that contains the correct answer. This mark is later printed on the instructor's copy of the test and greatly facilitates the marking of the key card. Of course, the mark does not appear on the student's copy. The punching of the cards can be done either by a card punch machine or by a Friden flexowriter which produces a paper tape that can later be converted into card form.
2. A card that contains an 11 punch in column 7 is placed between each test question to signal to the computer the end of that question group.
3. The cards from step 2 are then processed by the computer using option zero (0) of the test writing program. This process creates a new file of cards and, in addition to copying columns 11-80 of the input cards,

consecutively numbers the cards of each test question while punching a unique number for each question. During this same processing, a two digit code is punched into all cards, which is used to identify the subject of the test questions. The end of the question card (11 punch in 7) is deleted at this time and the identification punch is transferred to column 1 of the last card of each question group.

4. The output from step 3 forms the master processing file and is now used to do the actual test prepatation. A listing of this output is prepared showing both the unique number for each question and the subject code. The question and subject code for each desired question is recorded by the instructor and turned over to the machine operator. After loading the program, and selecting the correct program option (3), the operator merely enters the desired question numbers into the computer by the typewriter and the computer then produces the selected questions sequentially numbered and in a form which can be listed easily on the IBM 1443 printer or IBM 407 accounting machine. The test questions can be selected in different sequences in order to produce multiple forms of the same test if the classes are large enough to require this method of control.

THE TEST WRITING PROGRAM PROVIDES THE FOLLOWING OPTIONS

OPTION 0 (Zero) -- Construction of the Original File
This option converts the cards from the format as outlined in part 1 of the preceding section into the form for use in processing step 3.

OPTION 1 -- Update the File Produced by Option 0
Questions may be deleted or added to the processing file. Any questions to be added must be in the same card format as the processing file.

OPTION 2 -- Convert File to Paper Tape
Option 2 allows the processing file to be punched on paper tape to provide a duplicate record of the test question file as a protection against loss or destruction.

OPTION 3 -- Test Writing Program
The computer selects from the processing file those questions that the operator enters from the typewriter. This is the option that prepares the test for use in the classroom.

OPTION 4 -- Goof Switch
Console switch 4 is used to allow the operator to correct any erroneous information entered into the computer program through the typewriter.

TEST WRITING PROGRAM -- CARD FORMAT OF PROCESSING DECK

1-3 Question number
4-5 Subject code
6-7 Sequence number within question
8 Department code
9 This digit identifies all cards that belong to one part of the question. All cards that comprise the question contain a zero. Those belonging to response 1 contain a 1 punch, etc.
11-80 Contain an exact duplication of the input cards for Option 0.

1-1. WHICH OF THE FOLLOWING IS NOT A FUNCTION OF TOP MANAGEMENT IN A DATA PROCESSING UNDERTAKING.

1. APPROVAL
2. DIRECTION
3. CO-ORDINATION
- 4. OPERATION

2-1. THE DATA PROCESSING MANAGER OF A LARGE EDP INSTALLATION WOULD MOST LIKELY HAVE AS A JOB TITLE

1. TAB SUPERVISOR
2. TAB OPERATOR
- 3. CO-ORDINATOR
4. TAPE JOCKEY

3-1. ANALYSIS OF SYSTEMS, PROGRAMMING, LIAISON, PROCEDURES AND DOCUMENTATION, AND CONVERSION REPRESENT

- 1. PLANNING AND DEVELOPMENT FUNCTIONS
2. OPERATIONAL FUNCTIONS

ANY COMMENT CAN BE ENTERED AFTER A GIVEN QUESTION THRU THE TYPEWRITER

4-1. WHICH OF THE FOLLOWING GROUPS SHOULD RECEIVE NO TRAINING OR INDOCTRINATION PRIOR TO THE INSTALLATION OF AN EDP SYSTEM.

1. MANAGEMENT
2. PROGRAMMERS AND SYSTEMS ANALYSTS
3. OTHER EMPLOYEES
- 4. NONE OF THE ABOVE

5-1. THE COMPANY INSTALLING AN EDP SYSTEM WOULD PROBABLY NOT KEEP

1. A GENERAL PRE-INSTALLATION SCHEDULE
2. AN APPLICATIONS DEVELOPMENT SCHEDULE
- 3. A COMPUTER MANUFACTURING PROGRESS SCHEDULE
4. A PROGRAM DEVELOPMENT SCHEDULE

6-1. AN ASPECT OF PREPARATION FOR AN EDP SYSTEM WHICH IS MOST FREQUENTLY OR EASILY SLIGHTED IS

1. PROGRAMMING
2. CONVERSION
3. TESTING
- 4. DOCUMENTATION

7-1. THE PUNCHED CARD METHOD OF ACCOUNTING WAS DEVELOPED ABOUT

- 1. 1886
2. 1765
3. 1916
4. 1936

WHICH OF THE FOLLOWING IS NOT A FUNCTION OF TOP MANAGEMENT IN A DATA PROCESSING UNDERTAKING.

1. APPROVAL
2. DIRECTION
3. CO-ORDINATION
4. OPERATION

THE DATA PROCESSING MANAGER OF A LARGE EDP INSTALLATION WOULD MOST LIKELY HAVE AS A JOB TITLE

1. TAB SUPERVISOR
2. TAB OPERATOR
3. CO-ORDINATOR
4. TAPE JOCKEY

ANALYSIS OF SYSTEMS, PROGRAMMING, LIAISON, PROCEDURES AND DOCUMENTATION, AND CONVERSION REPRESENT

1. PLANNING AND DEVELOPMENT FUNCTIONS
2. OPERATIONAL FUNCTIONS

WHICH OF THE FOLLOWING GROUPS SHOULD RECEIVE NO TRAINING OR INDOCTRINATION PRIOR TO THE INSTALLATION OF AN EDP SYSTEM.

1. MANAGEMENT
2. PROGRAMMERS AND SYSTEMS ANALYSTS
3. OTHER EMPLOYEES
4. NONE OF THE ABOVE

THE COMPANY INSTALLING AN EDP SYSTEM WOULD PROBABLY NOT KEEP

1. A GENERAL PRE-INSTALLATION SCHEDULE
2. AN APPLICATIONS DEVELOPMENT SCHEDULE
3. A COMPUTER MANUFACTURING PROGRESS SCHEDULE
4. A PROGRAM DEVELOPMENT SCHEDULE

AN ASPECT OF PREPARATION FOR AN EDP SYSTEM WHICH IS MOST FREQUENTLY OR EASILY SLIGHTED IS

1. PROGRAMMING
2. CONVERSION
3. TESTING
4. DOCUMENTATION

THE PUNCHED CARD METHOD OF ACCOUNTING WAS DEVELOPED ABOUT

1. 1886
2. 1765
3. 1916
4. 1936

THE FIRST PUNCHED CARD APPLICATION WAS

1. PAYROLL
2. STATISTICS
3. INVENTORY
4. ADDRESSING

0019901X0 WHICH OF THE FOLLOWING IS NOT A FUNCTION OF TOP MANAGEMENT IN A DATA
0019902X0 PROCESSING UNDERTAKING.
0019903X1 1. APPROVAL
0019904X2 2. DIRECTION
0019905X3 3. CO-ORDINATION
-019906X4 - 4. OPERATION

0029901X0 THE DATA PROCESSING MANAGER OF A LARGE EDP INSTALLATION WOULD MOST
0029902X0 LIKELY HAVE AS A JOB TITLE
0029903X1 1. TAB SUPERVISOR
0029904X2 2. TAB OPERATOR
0029905X3 - 3. CO-ORDINATOR
-029906X4 4. TAPE JOCKEY

0039901X0 ANALYSIS OF SYSTEMS, PROGRAMMING, LIAISON, PROCEDURES AND
0039902X0 DOCUMENTATION, AND CONVERSION REPRESENT
0039903X1 - 1. PLANNING AND DEVELOPMENT FUNCTIONS
-039904X2 2. OPERATIONAL FUNCTIONS

0049901X0 WHICH OF THE FOLLOWING GROUPS SHOULD RECEIVE NO TRAINING OR INDOC-
0049902X0 TRINATION PRIOR TO THE INSTALLATION OF AN EDP SYSTEM.
0049903X1 1. MANAGEMENT
0049904X2 2. PROGRAMMERS AND SYSTEMS ANALYSTS
0049905X3 3. OTHER EMPLOYEES
-049906X4 - 4. NONE OF THE ABOVE

0059901X0 THE COMPANY INSTALLING AN EDP SYSTEM WOULD PROBABLY NOT KEEP
0059902X1 1. A GENERAL PRE-INSTALLATION SCHEDULE
0059903X2 2. AN APPLICATIONS DEVELOPMENT SCHEDULE
0059904X3 - 3. A COMPUTER MANUFACTURING PROGRESS SCHEDULE
-059905X4 4. A PROGRAM DEVELOPMENT SCHEDULE

0069901X0 AN ASPECT OF PREPARATION FOR AN EDP SYSTEM WHICH IS MOST FREQUENTLY
0069902X0 OR EASILY SLIGHTED IS
0069903X1 1. PROGRAMMING
0069904X2 2. CONVERSION
0069905X3 3. TESTING
-069906X4 - 4. DOCUMENTATION

0079901X0 THE PUNCHED CARD METHOD OF ACCOUNTING WAS DEVELOPED ABOUT
0079902X1 - 1. 1886
0079903X2 2. 1765
0079904X3 3. 1916
-079905X4 4. 1936

0089901X0 THE FIRST PUNCHED CARD APPLICATION WAS
0089902X1 1. PAYROLL
0089903X2 - 2. STATISTICS
0089904X3 3. INVENTORY
0089905X4 4. ADDRESSING

DATE 04/16/64 MIAMI-DADE JUNIOR COLLEGE
SEQ 4172 COMPUTER LABORATORY

STUDENT NUMBER	STUDENT NAME	NUMBER RIGHT	NUMBER WRONG	NO ANS	PER-CENT
0298325		52	48	0	52
0760714		25	56	19	25
1019448		48	49	3	48
1605231		29	67	4	29
1647911		39	61	0	39
1800106		65	34	1	65
2343836		77	23	0	77
2417329		63	36	1	63
2618043		40	58	2	40
2695433		51	48	1	51
2721826		68	32	0	68
3242714		53	43	4	53
3296833		79	21	0	79
3646443		56	44	0	56
3711731		71	28	1	71
3715941		46	52	2	46
3726436		34	62	4	34
3728531		57	42	1	57
4105026		44	55	1	44
4184543		54	45	1	54
4573521		37	62	1	37
4756026		40	60	0	40
5002821		67	31	2	67
0047206		53	45	2	53
0288421		51	48	1	51
0602041		58	40	2	58
1024806		33	66	1	33
1416711		58	42	0	58
1537641		37	60	3	37
1760831		43	56	1	43
1822026		75	22	3	75
2104443		58	40	2	58
2164121		65	35	0	65
2730223		43	55	2	43
2763631		50	46	4	50
2783706		55	44	1	55
3051010		37	54	9	37
3053426		45	55	0	45
3278106		59	37	4	59
3407846		53	45	2	53
3578216		44	56	0	44
3604023		75	23	2	75
3771711		67	33	0	67
3791611		45	55	0	45
4370211		73	26	1	73
4383133		64	35	1	64
4834426		71	26	3	71
4856606		53	42	5	53
5103131		70	29	1	70

DATE 04/16/64
SEQ 4177

MIAMI-DADE JUNIOR COLLEGE
COMPUTER LABORATORY

		STUDENT NUMBER	STUDENT NAME	NUMBER RIGHT	NUMBER WRONG	NO ANS	PER-CENT
ER	4	*0042407	STUDENT NUMBER INVALID	0	0	0	0
ER	4	*0042407	STUDENT NUMBER INVALID	0	0	0	0
		0070307		62	38	0	62
		0265938		71	29	0	71
		0488334		65	34	1	65
		0719341		55	43	2	55
		1021817		69	29	2	69
		1112831		66	32	2	66
ER	4	1142226		0	0	0	0
		1210041		60	39	1	60
		2748816		62	37	1	62
		3328636		53	47	0	53
ER	3	3365216		68	31	1	68
		3708436		69	28	3	69
		3899131		65	35	0	65
		4037106		37	45	18	37
		4267931		55	45	0	55
		4429911		68	32	0	68
		4538431		47	52	1	47
ER 2	4	5054811		0	0	0	0
		5134935		54	45	1	54
		5156246		64	36	0	64

DATE 04/16/64 MIAMI-DADE JUNIOR COLLEGE
COMBINED SEQUENCES 41XX COMPUTER LABORATORY

ITEM	N/A	1	2	3	4	5	R
1.	4	123	692*	76	457	14	0.155
2.	7	474*	359	305	217	4	0.209
3.	6	542	549*	56	132	81	0.130
4.	9	458*	212	130	322	235	0.330
5.	7	342	102	287	162	466*	0.245
6.	5	332	152	26	841*	10	0.346
7.	0	993*	286	28	25	34	0.410
8.	5	197	44	7	10	1103*	0.278
9.	3	338*	940	30	27	28	0.159
10.	3	92	1049*	140	48	34	0.347
11.	3	140	29	27	1141*	26	0.316
12.	5	57	68	86	21	1129*	0.331
13.	2	31	13	40	1070*	210	0.255
14.	7	655	478*	70	73	83	0.279
15.	7	117	205	68	885*	84	0.260
16.	10	508*	225	272	113	238	0.276
17.	1	184	641*	55	20	465	0.266
18.	2	114	292	867*	78	13	0.311
19.	2	895*	249	130	77	13	0.314
20.	3	120	814*	64	88	277	0.315
21.	4	144	934*	103	72	109	0.299
22.	5	565	174	222	315*	85	0.135
23.	7	294	97	93	773*	102	0.282
24.	3	66	520*	417	98	262	0.154
25.	10	34	48	21	120	1133*	0.224
26.	6	79	341	810*	57	73	0.250
27.	24	377	376	450*	106	33	0.294
28.	8	88	60	471	697*	42	0.162
29.	10	97	45	44	1087*	83	0.198
30.	5	90	155	42	1023*	51	0.229
31.	11	892*	40	161	245	17	0.339
32.	3	277	269	570*	39	208	0.233
33.	4	181	97	374	698*	12	0.290
34.	3	291*	57	27	617	371	0.333
35.	22	91	1102*	32	27	92	0.241
36.	2	270	57	257	598*	182	0.403
37.	8	1030*	153	71	26	78	0.424
38.	380	180	316	187	254*	49	0.071
39.	29	334	452*	305	160	86	0.421
40.	15	175	391	91	116*	578	0.051
41.	23	92	1043*	78	30	100	0.269
42.	10	136	101	134	640	345*	0.330
43.	11	583	47	43	342	340*	0.349
44.	72	22	343	289	148	492*	0.379
45.	178	115	394	14	644*	21	0.105
46.	26	445*	101	51	380	363	0.227
47.	11	267	690*	98	137	163	0.227
48.	1	191	51	567	547*	9	0.318
49.	1	18	22	967*	43	315	0.203
50.	6	413	154	29	683*	81	0.315

DATE 04/16/64 MIAMI-DADE JUNIOR COLLEGE
COMBINED SEQUENCES 41XX COMPUTER LABORATORY

ITEM	N/A	1	2	3	4	5	R
51.	2	51	47	1202*	63	1	0.188
52.	2	82	1094*	73	47	68	0.355
53.	4	776	135	253*	104	94	0.239
54.	10	258	401*	235	67	395	0.245
55.	4	68	179	162	765*	188	0.406
56.	2	100	79	209	784*	192	0.317
57.	2	229	136	117	196	686*	0.413
58.	3	354	70	88	109	742*	0.410
59.	11	176	338*	632	147	62	0.132
60.	8	202	276	166	663*	51	0.304
61.	7	395	133	709*	21	101	0.293
62.	4	61	56	548	585*	112	0.077
63.	5	234	171	737*	112	107	0.376
64.	1	409	250	67	628*	11	0.238
65.	7	864*	237	143	108	7	0.401
66.	1	17	35	1074*	41	198	0.281
67.	3	154	97	75	924*	113	0.379
68.	5	182	913*	176	88	2	0.267
69.	8	756*	158	206	92	146	0.196
70.	6	252	103	74	218*	713	0.077
71.	5	429	455*	113	24	340	0.329
72.	1	89	118	953*	160	45	0.363
73.	4	29	150	844*	256	83	0.426
74.	4	141	173	29	10	1009*	0.259
75.	11	319*	114	311	551	60	0.352
76.	7	194	1001*	15	25	124	0.369
77.	30	504*	260	75	97	400	0.117
78.	11	200	702*	179	228	46	0.166
79.	11	588*	94	268	135	270	0.373
80.	5	14	556*	108	136	547	0.321
81.	21	187	832*	118	169	39	0.227
82.	5	347*	58	463	377	116	0.250
83.	5	66	1090*	45	81	79	0.264
84.	3	438	128	643*	134	20	0.266
85.	13	881*	35	323	58	56	0.353
86.	4	163	663	144	215	177*	0.018
87.	12	426*	393	233	295	7	0.232
88.	405	257*	143	184	211	166	0.309
89.	24	591*	432	28	251	40	0.301
90.	15	171	368	366	392*	54	0.166
91.	29	63	395	699*	158	22	0.339
92.	11	111	770*	110	310	54	0.275
93.	10	51	17	203	1034*	51	0.184
94.	41	918*	63	94	65	185	0.290
95.	128	27	159	998*	37	17	0.304
96.	34	497*	127	305	180	223	0.198
97.	5	294*	250	186	299	332	0.267
98.	4	179	163	865*	102	53	0.274
99.	4	584	246	214	19	299*	0.265
100.	6	103	70	1033*	119	35	0.339

MEAN- 50.982
MEDIAN- 51
STANDARD DEVIATION- 12.544
SUM 69642

NUMBER OF QUESTIONS = 100
NUMBER OF TESTS GRADED = 1366
SUM OF SQUARES 3765338

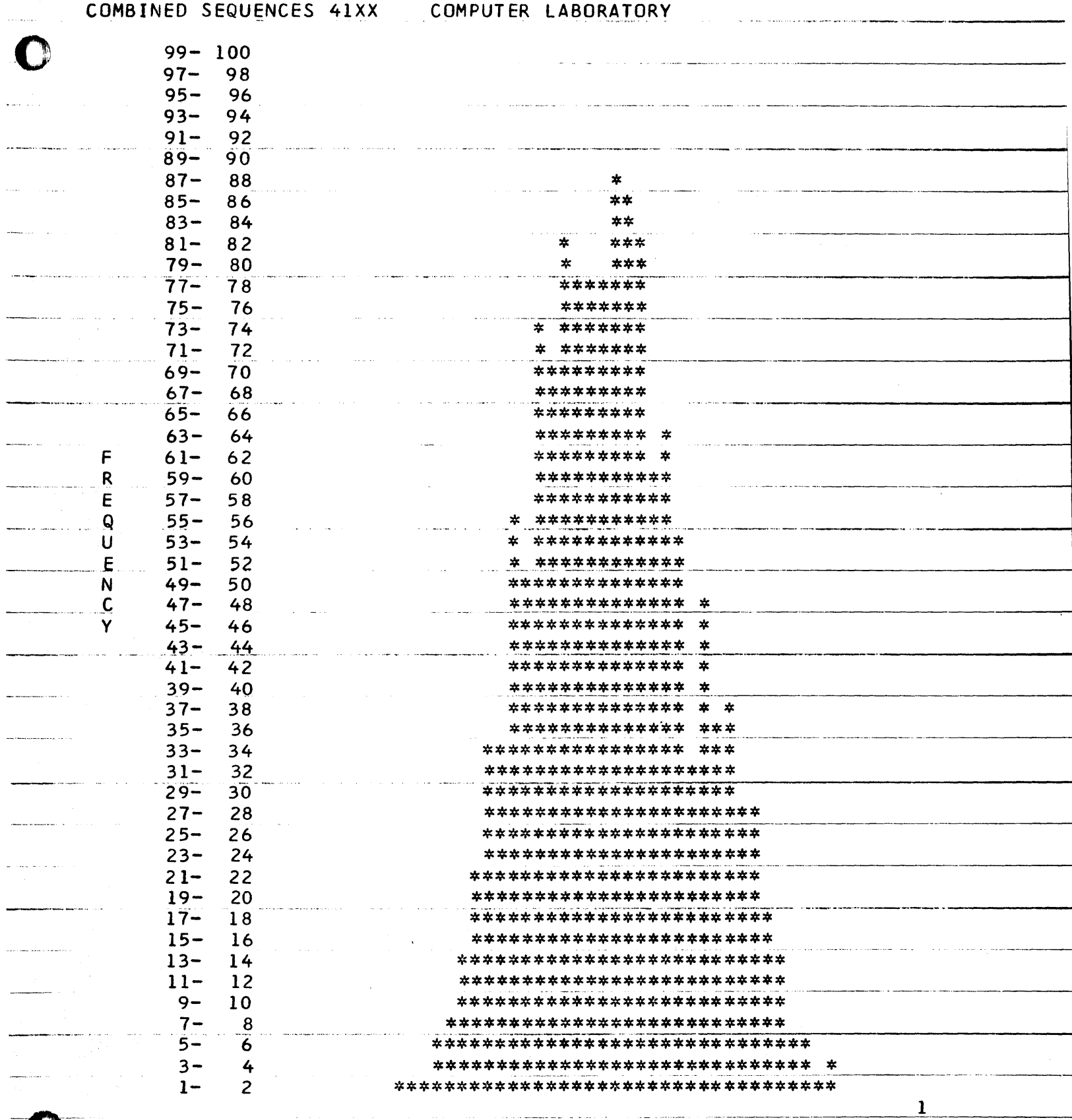
DATE 04/16/64 MIAMI-DADE JUNIOR COLLEGE
COMBINED SEQUENCES 41XX COMPUTER LABORATORY

DISTRIBUTION

SCORE	FREQ	CUM FREQ
85.	4	4
84.	1	5
82.	3	8
81.	3	11
80.	3	14
79.	3	17
78.	7	24
77.	7	31
76.	9	40
75.	9	49
74.	12	61
73.	15	76
72.	10	86
71.	17	103
70.	18	121
69.	19	140
68.	17	157
67.	19	176
66.	19	195
65.	28	223
64.	14	237
63.	17	254
62.	33	287
61.	21	308
60.	28	336
59.	35	371
58.	35	406
57.	24	430
56.	40	470
55.	41	511
54.	46	557
53.	40	597
52.	35	632
51.	53	685
50.	40	725
49.	37	762
48.	48	810
47.	29	839
46.	44	883
45.	33	916
44.	32	948
43.	49	997
42.	38	1035
41.	32	1067
40.	34	1101
39.	39	1140
38.	24	1164
37.	26	1190
36.	27	1217
35.	29	1246

SCORE	FREQ	CUM FREQ
34.	18	1264
33.	16	1280
32.	21	1301
31.	12	1313
30.	12	1325
29.	9	1334
28.	9	1343
27.	5	1348
26.	2	1350
25.	6	1356
24.	3	1359
23.	2	1361
22.	1	1362
21.	1	1363
20.	1	1364
19.	1	1365
17.	1	1366

DATE 04/16/64 MIAMI-DADE JUNIOR COLLEGE
COMBINED SEQUENCES 41XX COMPUTER LABORATORY



1. *BE WELL PREPARED* - Your entire paper should have been completed well in advance of the meeting. This will allow sufficient time for having the manuscript completely typed to be turned in to the Program Chairman at the meeting. In addition, all supporting materials such as slides and transparencies should be prepared and checked before your presentation. Examples, if used, should be chosen to best illustrate the points which you wish to make.

2. *BE AS CONCISE AS POSSIBLE* - Your purpose in presenting the talk is to convey information to your listeners. Anything extra is probably unnecessary. Your criterion should be whether or not the added material will contribute to this transfer of information. In particular, long stories involving personal experiences and long jokes should be kept for bull sessions.

3. *SPEAK CLEARLY AND DISTINCTLY* - In order to be understood, you must be heard! To be heard, you must face the audience and speak directly to them. Good eye contact is necessary. Your presentation should be at a rate slow enough to be clearly understood, and with sufficient volume to be heard throughout the room. If a microphone is available, use it and speak directly into the mike. Try to keep the distance between you and the mike constant. Varying this distance will cause a variation in the volume as heard by the listener.

4. *USE THE BLACKBOARD AS LITTLE AS POSSIBLE* - The blackboard is generally unsatisfactory for conveying information to the listener. If the blackboard is used at all, several pitfalls should be avoided. The speaker should take care not to talk while facing the blackboard. He should also take care to write large enough and clearly enough so that the material can be read all over the room. This is especially critical in large rooms. If much material is to be illustrated, it should not be put on the blackboard at all. Slides and transparencies are much more satisfactory for illustrating parts of your presentation.

5. *SLIDES AND TRANSPARENCIES SHOULD BE WELL PREPARED* - A good presentation can be ruined by sloppy slides! The information on the slide should be readable from the rear of a fairly large room. This is particularly true with respect to slides used for presenting papers at general sessions. Too many people make the mistake of including too much material on a slide. Remember, visual aids are used to explain or illustrate certain specific points in your talk--not to give a visual display of the entire talk. Remember that a criterion here is to have only those points on slides which will be directly referred to and will be needed for clarifying your presentation. It is very disconcerting to the listener to have slides cluttered with information and only one or two references made to the material contained thereon.

6. *USE HANDOUTS LIBERALLY* - In spite of our impressions as to how good our presentations are, it is still very true that the spoken word is soon forgotten. In particular, when light conditions are low, it is difficult for listeners to take notes. Written material in the form of handouts generally serves the purpose of making your talk remembered and of allowing people to have reference to such material at a later time. The handout can be as lengthy as you

wish it to be and can include the raft of material which should be left out of your slides and your oral presentation. In fact, additional examples are often helpful in handouts. If such handouts are prepared, it might be wise to include the mailing address where the author or authors can be contacted if additional information is needed.

7. *STICK TO ALLOTTED TIME* - Time is valuable and all papers must be presented! By means of one or more rehearsals, you should be well aware of the time required for your presentation. It should be of such a length that you will complete it two or three minutes ahead of the scheduled time. In that way, a few questions can be asked by members of the audience. *IN NO CASE SHOULD YOU TALK LONGER THAN THE TIME ALLOTTED FOR YOUR PAPER.* Remember, when you exceed your time limit, you are depriving someone else of the opportunity to make his presentation.

8. *WRITTEN REPORT* - For the purpose of reproducing your report in the Proceedings, have your paper typed single spaced. We have a limited amount of space; in order to present the papers and by being careful, the space can be well used. Do not include long program listings and long examples in your paper. It should, in no case, be longer than the oral presentation which you made.

Carol A. Hall
Program Chairman

Roster of Attendants

Atkins, Daniel E.	Bucknell University	Lewisburgh, Pa.
Austin, George A.	C.W. Post College	Greenvale, N.Y.
Baber, Gaye M.	Nat'l Education Assoc.	Washington, D.C.
Baron, Ray	Raytheon Co.	Bedford, Mass.
Beaudreau, Miss Doreen	Univ. Of Wisconsin	Milwaukee, Wisconsin
Best, A.H.	The Glidden Co.	Jacksonville, Fla.
Billen, Geoffrey G.	Clarkson College	
Bleuel, Wm. H.	General Dynamics	Rochester, N.Y.
Bras, Rodney	I.B.M.	White Plains, N.Y.
Bridgeman, Keene A.	General Dynamics/Electronics	Rochester, N.Y.
Brown, Leverett W.	Southern Div. of Int'nl Paper	Mobile, Ala.
Bryant, Charles W.	U.S. Navy	Crane, Indiana
Bryan, Norman W.	Computer Service, Inc.	Jacksonville, Fla.
Callahan, Elias R., Jr.	Chattanooga St. Tech. Inst.	Chattanooga, Tenn.
Cambris, Joseph T.	Murray On Corporation	Meraux, La.
Carroll, Everette L.	Tennessee Highway Dept.	Nashville, Tenn.
Caslin, James	Aerospace Research Lab.	Wright Paterson A.F. Base, Ohio
Caudill, Edward Charles	Cincinnati Milling Mach. Co.	Cincinnati, Ohio
Champagne, Wiltt P.	Southeastern La. College	Hammond, La.
Channen, E.W.	University of Windsor	Windsor, Ontario, Canada
Chaverin, Carl L.	I.B.M.	Chicago, Illinois
Chow, Dr. Wen M.		Princeton, New Jersey
Chretien, Mas	Brandeis University	Waltham, Mass.
Cicchinelli, Dr. A.L.	Clarkson College	Potsdam, N.Y.
Clark, Henry T.	Raytheon Co.	Bedford, Mass.
Clarke, Reverend Arthur	Fordham University	Bronx, N.Y.
Closman, S.S.	I.B.M.	White Plains, N.Y.
Czyzewski, L.J.	De Leuw, Cather & Co.	Chicago, Illinois
Davidson, Charles	University of Wisconsin	Madison, Wisconsin
Dibeneditto, Michael	New Departure-Hyott,Div. GMC	Waterbury, Connecticut
Dole, Marilyn	Colorado State University	Fort Collins, Colorado
Dunham, Peter C.	Raytheon Co.	Wayland, Mass.
Dye, David R.	I.B.M.	White Plains, N.Y.
Eikeland, Marion D.	Tampa Electric Co.	Tampa, Fla.
Einhorn, Beth	Leesona Meos Laboratories	Great Neck, L.I.
Ellis, Raymon D.L.	Travelers Research Center	Hartford, Connecticut
Elrod, J.C.	Georgia Experiment Station	Griffin, Georgia
Ely, Arthur L.	Miami Dade Jr. College	Miami, Fla.
Farrell, Edward G.	Cooper Union Sch. of Eng.&Sci.	Cooper Sq., N.Y.
Ferling, John A.	Claremont Men's College	Claremont, Calif.
Fodor, Mrs. Joyce	University of Wisconsin	Madison, Wisconsin
Folse, Paul D.	Tampa Electric Company	Tampa, Florida
Freeman, Roy E.	Jr. College of Broward College	Hollywood, Fla.
Friedwald, Elliott C.	Electronic Data Processing Inc	Ft. Lauderdale, Fla.
Fuchs, Carole	Leesona, Moos Laboratories	Great Neck, L.I.
Furlott, Sherry M.	Raytheon Co.	Wayland, Mass.
Galle, Miss Janice	University of Texas Med. Br.	Galveston, Texas
Garcia, Oscar N.	Old Dominican College	Old Dominican College
Gary, Patricia	Electronic Data Processing Inc	Ft. Lauderdale, Fla.
Gelsi, Hector P.		New York, New York

George, Dr. Edward Y.	Bently College	Lexington, Mass.
Goldberg, Marvin		
Goldman, Norman	Boston Univ. Computing Center	Boston,Mass.
Gross, Donald S.	University of Maryland	College Pk., Md.
Grove, Richard E.	Randolph-Macon College	Ashland, Va.
Guillermety, Herman L.	Engineers and Architects	San Juan, Puerto Rico
Haag, James N.	Purdue University	Indianapolis, Indiana
Haas, I. John	Christion Brothers College	
Hale, John S.	State Univ. of N.Y. at Buffalo	Hoboken, New Jersey
Hall, Mrs. Carol A.	Univ. of Southwestern La.	Lafayette, La.
Hamilton, E. Michael	Human Resources Research Off.	Ales, Va.
Hartz, Theodore M. and Young, Susannah H.	Public Health Service	Rockville, Md.
Hatfield, Fred A.	Line Material Industries	Zanesville, Ohio
Heckart, Lona	Aerospace Corporation	Merritt Island, Fla.
Hoffman, Lanny	Princeton University	Highstown, New Jersey
Housman, Arthur C.	Royal Typewriter Co.	Hartford, Connecticut
Herring, Lt. Col. O.L.Jr.	The Citadel	Charleston, S.C.
Hester, Richard L.	Electric Water Engineer Dept.	Jacksonville, Fla.
Humbrecht, Robert V.	Schutte&Koerting Co.	Levittown, Pa.
Isbell, Robert G.	Ingalls Iron Works	Birmingham, Ala.
Ivester, Robert D.	Southern Engin. Co. of Georgia	Atlanta, Ga.
Johnson, Alan D.	NASA	Sandusky, Ohio
Johnson, Arthur	Royal Typewriter	
Jones, James L.	Girdler Corporation	Louisville, Kentucky
Jones, Robert L.	University of Maryland	Baltimore, Maryland
Jutsum, P.J.	U.W.I.	Kingston, Jamaica
Kemp, Jerry L.	U.S. Navy	Loogootee, Indiana
Kenney, Peter J.	Western New England College	Springfield, Mass.
Kerr, H.B.	Tennessee Tech	Cookeville, Tennessee
Kick, Russell C., Jr.	Stetson University	Daytona Beach, Fla.
Kien, Dr. Gerald A.	Northwestern Medical School	Chicago, Illinois
Kindred, Alton R.	Manatee Jr. College	Bradenton, Fla.
Knebel, Martin	Burns & Roe, Inc.	New York, New York
Korelitz, Ted	The Badger Co., Inc.	Cambridge, Mass.
Kubie, J.J.	I.B.M.	Gainesville, Fla.
Kuo, Dr. S. S.	University of New Hampshire	Durham, New Hampshire
Kenngott, Robert L.	Charles H. Sells, Inc.	Pleasantville, New York
Lang, Andrew J.	Fairchild DuMont Labs	Clifton, New Jersey
Larrea, Pablo	Princeton University	Somerville, New Jersey
Leno, Morris E.	NYS Dept. of Pub. Works	Albany, New York
Li, Yuling	Harvard Medical School	Boston, Mass.
Lilly, G.F.	Jones & Langhlin Steel Corp.	Pittsburgh, Pa.
Little, Frank S.	Orlando Utilities Commission	Orlando, Fla.
MacKenna, Mr. Craig	University of Wisconsin	Milwaukee, Wisconsin
MacMullin, R.Bruce	Western Supply Co.	Tulsa, Oklahoma
Madden, James W.	Electronic Data Processing	Ft. Lauderdale, Fla.
Markland, Thomas I.	I.B.M.	
Marks, Maxwell	I.B.M.	White Plains, N.Y.
Maskiell, Frank H.	Penna. Trans. Div. McGraw Edison	Canonsburg, Pa.
Maudlin, Charles	University of Oklahoma	
McGahey, Patricia E.	Pratt-Whitney Aircraft	West Palm Beach, Fla.

McGehee, Lcdr Thomas L.	U.S. Navy	Pensacola, Fla.
Meister, Wm. R.	Philip Morris Inc.	Richmond, Va.
Mello, Leonard	Geophysics Corp. of America	Woburn, Mass.
Meyer, R.H.	De Leuw, Cather & Company	Chicago, Illinois
Meyer, Mr. Rudolf	St. Univ. of N.Y. at Buffalo	Buffalo, N.Y.
Miller, Mrs. Ruth B.	General Motors Corp.	Jasport, N.Y.
Mishelof, Richard	Downstate Medical Center	Brooklyn, New York
Mowchan, Michael	Monsanto Research	Dayton, Ohio
Newton, Jr., Lawrence E.	Univ. of Texas	Houston, Texas
Niceley, John B.	W.W. Holding Ind. Ed. Center	Raleigh, N.C.
Oliver, James R.	Univ. of Southwestern La.	Lafayette, La.
Owen, David G.	Miami-Dade Jr. College	Miami, Fla.
Pachou, Heberto	Automatic Electric Labs	Northlake, Illinois
Parker, J. Leslie	I.B.M.	Poughkeepsie, N.Y.
Parker, Robert H.		Ottawa, Ontario, Canada
Parsons, G. Linwood,Jr.	McGaughy, Marshall&McMillan	Norfolk, Virginia
Peeples, Carl B.	Electrical Engin. Dept.	Jacksonville, Fla.
Plesums, Charles A.	Union College	Schenectady, New York
Prater, Merle	I.B.M.	Endicott, New York
Ramsen, J.A.	I.B.M. University Program	Chicago, Illinois
Raver, Richard E.	W.R. Grace & Company	Highland, Maryland
Remilen, Charles H.	Eastman Kodak Company	Rochester, New York
Richter, I. Kenneth	Delaware St. Highway Dept.	Dover, Delaware
Rivet, Henry	New York State	Rensselaer, New York
Rock, Samuel	E.R. Squibb & Sons	Metuchen, New Jersey
Ross, Richard D.	University of Mississippi	
Ross, Tony A.	University of Mississippi	
Rudolphe, Mrs. Jean	U.S. Dept. of Agriculture	Washington, D.C.
Russell, James H.	Gilbert Associates, Inc.	Reading, Pa.
Salus, Larry	Electronic Data Processing, Inc	Ft. Lauderdale, Fla.
Scaletti, Henry M.	United Shoe Machine Corp.	Beverly, Mass.
Scarpato, Marie	Stevens Inst. of Technology	Hoboken, New Jersey
Scheeron, W.G.	Bell Telephone Laboratories	North Andover, Mass.
Scheinok, Dr. P.	Hahnemann Medical College	Phila., Pennsylvania
Schmalenberger, J.A.	Pan American World Airways	Cape Kennedy, Fla.
Schrodel, C.S.	Sun Oil Company	Phila., Pennsylvania
Scott, Edward E.	Lukens Steel Co.	Parkersburg, Pa.
Scott, I.J.	Sun Oil Co.	Phila., Pa.
Sekscienski, Wm.	University of Maryland	College Park, Maryland
Sinanian, Ed	I.B.M.	New York, New York
Sisson, John S., Jr.	E.I. Dupont de Nemours & Co.	Chattanooga, Tenn.
Smith, Bryan		Ottawa, Ontario
Smith, John A.	Stone & Webster Service Corp.	New York, New York
Spitalny, Arnold	Norden Div., United Aircraft	Norwalk, Connecticut
Stacke, Wanda B.	American Tel & Tel Co.	
Staiano, Edward F.	Bucknell University	Lewisburg, Pa.
Steele, Laura B.	Beneral Motors Institute	
Stone, Mrs. Eleanor D.	Brandeis University	Waltham, Mass.
Stubbe, John	Clark University	Worcester, Mass.
Sutton, Joseph T.	Stetson University	DeLand, Fla.

Sweatman, Arthur T.	Lukens Steel Co.	West Chester, Pa.
Taranto, Frank	EDO Corporation	College Point, N.Y., N
Thatcher, Charles M.	Pratt Institute,	Brooklyn, New York
Thayer, Raymond J.	Line Material Industries	Zanesville, Ohio
Thomas, Raymond E.	George Washington University	Washington, D.C.
Thwing, Henry W.	Stetson University	Deland Fla.
Trevino, Jose	Monterrey Inst. of Technology	Monterrey, Mexico
Tuttle, Dr. W.N.	General Radio Co.	Concord, Mass.
Voytovich, Miss Sharon	Syracuse Univ. Research Corp.	Syracuse, N.Y.
Wages, J.E.	Imperial Tabacco Co.	
Wall, H.M.	I.B.M.	Saxonville, Mass.
Watson, Jack B.	Texas Gulf Sulphur Co.	New Gulf, Texas
Wigdahl, Allen B.	Allen-Bradley Co.	Milwaukee, Wisconsin
Williams, C.R.	Dow Chemical Co.	Lake Jackson, Texas
Wilson, Charles R.	Hamden Testing Services	Montclair, New Jersey
Wingert, Joseph T.	Trenton Jr. College	Langhorn, Pa.
Wright, Donald L.	Georgetown University	Arlington, Va.
Wright, Lawrence	Sprague Electric Co.	North Adams, Mass.
Wuensch, Alfred	Columbia University	
Young, Barbara F.	Pan-American World Airways	Patrick A.F. Base, Fla.
Young, John W.	Radiation, Inc.	Melbourne, Fla.
Young, Mrs. Susannah H.	Public Health Service	Rockville, Maryland