Call for papers!

PRESIDENT
J. L. DAVIDSON
Long Island Lighting Co.
175 Old Country Road
Hicksville, New York

1620 USERS GROUP

DR. JOHN MANIOTES
COMPUTER TECHNOLOGY DEPT.
PURDUE UNIVERSITY
CALUMET CAMPUS
HAMMOND, IN 46323

Call for papers!

TREASURER
CARLSON
of Public Works
1220 Washington Avenue
Albany 26, New York

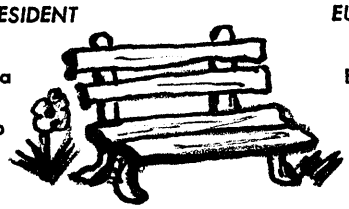
WESTERN REGION PRESIDENT
PAUL BICKFORD
Univ. of Okla. Medical Research
800 N. E. 13th Street
Oklahoma City, Oklahoma

+

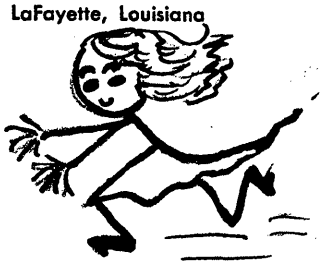
MID-WESTERN REGION PRESIDENT
W. A. BURROWS
Dravo Corporation
Neville Island
Pittsburgh 25, Pennsylvania

EASTERN REGION PRESIDENT
J. R. OLIVER
Univ. of Southwestern La.
Box 133, USL Station
LaFayette, Louisiana

CANADIAN REGION PRESIDENT
D. A. JARDINE
Dupont of Canada
Research Center
Kingston, Ontario



EUROPEAN REGION PRESIDENT
H. TOMPA
European Research Associates
95 Rue Gatti De Gamond
Bruxelles 18, Belgium



WE'RE JOINING OUR FORCES!

The Joint Western and Mid-Western Region Meeting will be held November 9, 10, and 11, 1964, at the Center for Continuing Education on the University of Oklahoma Campus, Norman, Oklahoma.

The following areas will be of particuliar interest:

1. PERT and Critical Path reporting systems
2. Operations Research
3. Numerical Control
4. Programming Procedures and Systems
5. Numerical Analysis, other Math, and Statistics
6. 1710 Applications, all areas
7. Management Games
8. Education; Administration, Teaching

Call for papers!

We need people for 1311 and Education panel members. Presently, there seems to be much interest in a customer-contributed advanced level Monitor I or II Workshop. Perhaps you would like to discuss a problem you have spent considerable time investigating but for which you have found no desirable solution. Possibly you have been probing into the 360 Systems and could give others an enlightening comparison between the 1620 and, for instance, the Model 30.

Plan now to contribute something from your experience and let us all benefit.

Send the following information: (a) title and author (b) company name and Users Code (c) time required for presentation (d) special equipment required (e) technical level of paper (f) audience for whom it is intended (g) a brief abstract of the content.

SEPTEMBER 1 is the deadline. Send to:

Mr. George Larcade
Chemical Research and Development
Halliburton Company
Duncan, Oklahoma

Call for papers!

F I N A L · A N N O U N C E M E N T

Plan now to attend the Joint Western, Mid-Western Fall Meeting which promises to be one of the best yet.

WHEN: November 9, 10, 11, 1964

WHERE: Center for Continuing Education on the O. U. Campus, Norman, Oklahoma

Parallel sessions are planned for the entire meeting with a variety of interesting topics being presented. A 1620 Model II with disks, printer and plotter and peripheral equipment (407-E8, sorter, 026) will be available at the meeting for program testing and demonstration.

REGISTER EARLY ! ! ! !

Pre-registration is essential since housing facilities are somewhat limited and transportation to and from the airport must be planned. On the pre-registration form the following information is necessary: (Airline name, flight number and time) for both arrival and departure. Transportation charges to the Center for Continuing Education is $3.00/ person for 4 or more people per cab. Otherwise $10.00 per cab one-way. By knowing the above information every effort will be made to schedule the most economical transportation to the Center for Continuing Education from the airport.

Pre-registration is $10.00; at the meeting it is $13.00. ATTEND - and help make this one of the most successful Joint meetings.

PRESIDENT
D. A. JARDINE
Dupont of Canada
Research Center
Kingston, Ontario

1620 USERS GROUP

SECRETARY-TREASURER
CHARLES E. CARLSON
N. Y. State Dept. of Public Works
1220 Washington Avenue
Albany 26, New York

WESTERN REGION PRESIDENT
PAUL BICKFORD
Univ. of Okla. Medical Research
800 N. E. 13th Street
Oklahoma City, Oklahoma

MID-WESTERN REGION PRESIDENT
F. H. Maskiell
Pennsylvania Transformer Co.
Box 330
Canonsburg, Pennsylvania

EASTERN REGION PRESIDENT
J. R. OLIVER
Univ. of Southwestern La.
Box 133, USL Station
LaFayette, Louisiana

CANADIAN REGION PRESIDENT
V. G. Smith
Univ. of Toronto-Elect. Eng. Dept.
Galbraith Building
Toronto 5, Ontario

EUROPEAN REGION PRESIDENT
H. TOMPA
European Research Associates
95 Rue Gatti De Gamond
Bruxelles 18, Belgium

WESTERN & MID-WESTERN REGIONS JOINT MEETING ANNOUNCEMENT NORMAN, OKLAHOMA NOVEMBER 9, 10, 11, 1964

AGENDA

GENERAL INTEREST SESSIONS

CONCURRENT SPECIAL INTEREST SESSIONS

EDUCATION AND MEDICAL PANELS

FORTRAN IV

OPERATIONS RESEARCH

STATISTICS

CRITICAL PATH

NUMERICAL ANALYSIS

LINEAR PROGRAMMING

SORT/MERGE

ELECTRICAL UTILITY TEAM

1620 MODEL II / DISKS / PRINTER / PLOTTER

COME TO WORK. LISTEN AND LEARN!

PROGRAM AGENDA

Sunday - November 8, 1964

7:00 P.M. Registration - Social Hour — Commons Dining Hall

Monday - November 9, 1964

8:00 Registration — East Concourse

8:30 Welcome --- Paul Bickford, Western Region President -- Forum Room
IBM Announcements --- Dick Williams
Use of DP Library --- Dave Dye

10:00 Coffee

10:30 Systems/360 NPL -- Jay Michtom (IBM) — Forum Room

12:00 Lunch

1:00 Electrical Utility Team — Meeting Room A3

	Session I Forum Room	Session II Meeting Room A
1:00	Linear Programming -- Dr. S. T. Parker (3107)	Continuous Steel Beam Analysis & Design -- Stanley W. Woods (3072)
1:30	Linear Programming (Continued)	Automated Steel Building Design System -- L. L. Bathurst (1164)
2:00	Computerized Economic Evaluation -- T. I. Markland (IBM)	The Ohio State 1620 Programming System -- James Shaffer (3023)
3:00	Coffee	
3:30	New Linear Programming -- Harry Muller (IBM)	Adminstration of a Short Course in FORTRAN -- J. R. Wright (3320)
3:50	New L. P. (Continued)	An Analog-to-Digital Conversion System for the IBM 1620, Model I -- W. Wilcoxson (5045)
4:40	New L. P. (Continued)	PACTULAS - A. Digital Analog Simulator for the IBM 1620 - R. O. Brennan (IBM)

5:00 Adjournment of Day's Sessions

7:30 P.M. New Users Session --- Paul Bickford -- Forum Room

8:00 P.M. Sound-Off Session --- C. E. Maudlin -- Forum Room

Tuesday - November 10, 1964

7:30	Late Registration	East Concourse
	Session I Forum Room	Session II Meeting Room A
8:00	Advanced FORTRAN II Techniques -- Lanny Hoffman (1177)	A Multivariate Analysis Package -- Rex L. Hurst (5103)
8:45	FORTRAN II (Continued)	STRAP with FORMAT -- W. Wilcoxson (5045)
9:30	FORTRAN II (Continued)	Stochastic Simulation of Populations in Natural Selection Problems -- Alice M. Brues (3082)
10:00	Coffee	
10:30	FORTRAN II (Continued)	Application of Difference Equations in Numberical Analysis -- John NcNamee (5171)
11:00	FORTRAN Flow Chart Program -- Robert E. Babcock (5058)	Difference Equations (Continued)
11:30	SUNDYNE Pump Quoting System -- Robert E. Babcock (5058)	Critical Path Method of Planning, Scheduling and Control for Construction Operations -- Jim S. Thompson
12:00	Luncheon	
	Keynote Address:	
	A Computer's Reminiscent Remarks - Dr. Archie M. Kahan, Ph.D Executive Director of the University of Oklahoma Research Institute	
1:30	SPS Tutorial and Workshop --	Richard Pratt - Meeting Room A2
1:30	The Use of 1620-1311 System in Blood Bank Inventory Control -- Stanley Shannon (5226)	1620 FORTRAN IV - James White(3086)
1:45	Numberical Control - Tom Woods	FORTRAN IV (Continued)
2:15	Louisiana Tech Information Retrieval System -- David A. Hake (3087)	Programing for Selective Assembly -- John F. Mahan (3070)
2:45	Information Retrieval (Continued)	Experiences with MISS LESS and MISS LESS DATING SYSTEM - R.E. Dillion (3327)
3:00	Coffee	

Tuesday - November 10, 1964 (Cont'd)

3:30 STUFF - 1620 Universal Function Fitter -- IBM

Medical Panel:
Chairman: Dr. Edward N. Brandt, Jr., M.D., Ph.D.
O.U. Medical Research Computer Center
Oklahoma City, Okla.

Members: Dr. John Gustafson
Iowa Methodist Hospital
Des Moines, Iowa

William F. Blose
Baylor University College of Medicine, Houston, Texas

Lawrence Newton
M. D. Anderson Hospital
Houston, Texas

5:00 Adjournment of Day's Session

7:00 P.M. Series of Four Movies: (1) Plotter, (2) Drafting, (3) Telstar, (4) Sketch-Pad

9:00 Executive Board Meeting

Wednesday - November 11, 1964

	Session I Meeting Room A	Session II Forum Room
8:00	Simulation the Businessman's Laboratory -- Bill Bryan(IBM)	Sort/Merge -- Vern Boyer (IBM)
10:00	Coffee	
10:30	Writing FORTRAN Subroutines in SPS for the 1620 Computer -- Roy J. Wepman (IBM)	Education Panel: Chairman: Dr. Richard V. Andree Chairman, Mathematics Dept. University of Oklahoma Members: Woodfin C. Garrett High School Teacher Heavenear, Oklahoma J. R. Oliver, Dean Graduate School Director, Computing Center University of Southwestern Louisiana

Members: Jesse Richardson
Senior Supervisor in
Education
Department of Education
Boston, Massachusetts

Ronald Turner, Director
of Data Processing
Lansing, Michigan
Public Schools

11:30 FORTRAN II-D Subroutines
-- Frank Mozina (IBM)

12:00 Lunch

1:00 Frequently Used Computer Techniques That Do Not Work
-- Richard V. Andree (5771)

Systems/360 Operation System --
John Giffin (IBM)

2:00 Use of 1620 for Typesetting
-- Bill Williams

2:45 IBM Reply to Sound-Off

Meeting Adjournment

SYNPOSIS OF PAPERS

The sessions to be presented are summarized as follows: Title, Author (where applicable), Synopsis, Type, and level of paper or workshop.

Monday, November 9

Session I

"Linear Programming" - S. Thomas Parker

An introduction to the concepts and methods of solving linear programming problems.

Type: General Interest Level: Elementary

"Computerized Economic Evaluation" - T. I. Markland

This paper describes a computerized technique for making investment decisions.

Type: General Interest Level: Intermediate

"New Linear Programming" - Harry Muller

Type: General Interest Level: Intermediate

Session II

"Continuous Steel Beam Analysis and Design" - Stanley W. Woods and James F. Gibbons

A program written to analyze and design continuous steel beam bridges. The program will handle from two to five spans and could be extended for machines with more than 20K memory.

Type: Civil Engineering Level: Intermediate to Advanced

"Automated Steel Building Design System" - L. L. Bathurst

This paper is intended for people interested in computer systems as a concept.

Type: General Interest Level: Elementary

"The Ohio State 1620 Programming System" - James Shaffer

A unified loader-assembler-complier system on a 20K.

Type: General Interest Level: Intermediate

"Administration of a Short Course in FORTRAN" - James R. Wright

A way of acquainting employees, not closely associated with computers, with the FORTRAN language to the point of being able to read and understand a program written for their use.

Type: General Interest Level: Elementary

"An Analysis-to-Digital Conversion System for the IBM 1620 Model I" - W. Wilcoxson and A. H. Jackson

Type: General Interest Level: Intermediate

"'PACTOLUS' A Digital Analog Simulator for the IBM 1620" - R. D. Brennan and H. Sano

PACTOLUS includes all standard analog elements plus many special function blocks.

Type: General Interest Level: Intermediate

Tuesday, November 10

Session I

"Advanced FORTRAN II Programming Techniques" - Lanny Hoffman

An evening session will be arranged if interest indicated.

Type: General Interest Level: Advanced

"FORTRAN Flow Chart" - Robert E. Babcock

Given a FORTRAN source deck, up to and including level FORTRAN II-D, the FORTRAN Flow Chart Program will automatically generate a logic diagram for the program.

Type: General Interest Level: Elementary

"SUNDYNE Pump Quoting System" - Robert E. Babcock and F. R. Bollenbach

A system of programs which accepts customer requirements and designs and selects the proper pump and components, plots predicted performance, prints specification and pricing sheet all in form which can be sent directly to the potential customer.

Type: Automated Design Engineer Level: Intermediate

"The Use of the 1620-1311 Systems in Blood Bank Inventory Control" - Stanley Shannon and Paul Schoch

The inventory problem of a large metropolitan blood bank is described which involves a multi-level system with stochastic supply and demand.

Type: Special Applications Level: Intermediate

"Numerical Control" - Tom Woods

A brief description of the relationship of computer job-oriented languages especially designed for numerical control application and the production process.

Type: General Interest Level: Intermediate

"Louisiana Tech Information Retrieval System" - David A. Hake and Jesse H. Poore, Jr.

An information retrieval system developed for small, specialized "libraries" of up to approximately five thousand documents.

Type: General Interest Level: Intermediate

"STUFF - 1620 Universal Function Fitter" - IBM

Type: General Interest Level: Intermediate

Session II

"A Multivariate Analysis Package" - Rex L. Hurst

A series of interlocking programs developed to solve a portion of multivariate problems.

Type: Statistics Level: Introductory

"STRAP With FORMAT" - W. Wilcoxson and J. Wohlever

A program that computes a sequence of multiple regression equations in a stepwise manner.

Type: Math and Statistics Level: Intermediate

"Stochastic Simulation of Populations in Natural Selection Problems" - Alice M. Brues

Simulation of genetic changes taking place in isolated populations using Monte Carlo methods.

Type: Statistics Level: Intermediate

"Application of Difference Equations to Numerical Analysis" - John McNamee

Type: General Interest Level: Intermediate

"Critical Path Method of Planning, Scheduling and Control for Construction Operations" - Jim S. Thompson

Historical development of the Critical Path Method and some of the many applications to construction work.

Type: General Interest Level: Intermediate

"1620 FORTRAN IV" - James White

Specification, implementation, and use of a new programming system.

Type: General Interest Level: Advanced

"Programming for Selective Assembly" - John F. Mahan

Describes the development of a technique for simulating the selective grouping of assembly components to attain dimensional uniformity.

Type: Quality Control Level: Intermediate

"Experiences with MISS LESS and MISS LESS DATING SYSTEM" - R. E. Dillion and L. V. Parent

PERT applied to construction schedules, material delivery and restraints.

Type: General Interest Level: Elementary

Medical Panel

Each panel member will discuss how the 1620 is used at their particular installation.

Type: General Interest Level: Elementary

Wednesday, November 11

Session I

"Simulation the Businessman's Laboratory" - Bill Bryan

Type: Operations Research Level: Intermediate

"Writing FORTRAN Subroutines in SPS for the 1620 Computer" - Roy J. Wepman

The coding, assembling and utilizing of SPS written subroutine and function subprograms in 1620 FORTRAN II or FORTRAN II-D.

Type: General Interest Level: Advanced

"FORTRAN II-D Subroutines" - Frank Mozina

Subroutines will include reread, transmittal of blocks of data to and from disc.

Type: General Interest Level: Intermediate

"Frequently Used Computer Techniques That Do Not Work" - Dr. Richard V. Andree

A discussion of half a dozen commonly used mathematical techniques which are apt to lead to erroneous results when used on a computer.

Type: General Interest Level: Intermediate

"The Use of the 1620 for Type Setting" - Bill Williams

The pilot use of a computer for printing production programmed to justify lines of type and hyphenated words, when necessary for justification.

Type: General Interest Level: Intermediate

Session II

"Sort/Merge" - Vern Boyer

Type: General Interest Level: Introductory

Education Panel

Panel members will discuss how the 1620 is used at their particular installation.

Type: General Interest Level: Elementary

"Systems/360 Operation Systems" - John Giffin

Type: General Interest Level: Elementary

Papers classed as elementary presuppose no background on the part of the audience. Sessions classified as intermediate presuppose some acquaintance with the subject matter.

Sessions classed as advanced are intended for persons with considerable experience in the field; questions of an elementary nature will be discouraged at these sessions.

COMMENT SHEET

Please fill in this sheet and place in the comment box at the registration desk. This information is needed so that meetings can be planned to meet your needs.

1. List topics you would like to hear at the next meeting:

2. List ways in which you feel meetings could be improved:

3. List topics you liked at this meeting:

1620 USERS GROUP
WESTERN & MID-WESTERN REGION

REGISTRATION FOR 1964 FALL MEETING
CENTER FOR CONTINUING EDUCATION
UNIVERSITY OF OKLAHOMA
NORMAN, OKLAHOMA

COMPANY_____ USERS GROUP NO._____

ADDRESS_____

CITY_____ STATE_____

Name of Registrant(s)_____

_____

_____

I (we) plan to attend the following sessions or workshops:

New Users_____ SPS Tutorial_____
Education Panel_____ Medical Panel_____ Fortran II_____

I (we) would be interested in exchanging information with other members on the following subjects: (Please circle the appropriate numbers)

01 Civil Engr.
02 Electrical Engr.
03 Mechanical, Aero-Space and Petroleum Engr.
04 Chemical
05 Power
06 Math and Statistics
07 Education-College
08 Education-High School
09 Education-Grade & Jr. High
10 Behavioral Science
11 Medicine
12 Analog & Hybrid Computers
13 Industrial Application & Automatic Control
14 Govt., Military & Transportation Application
15 Research
16 Operations Research, Pert, CPM

Please list on the back of this page topics of interest that are presented at conflicting times.

THIS MUST BE FILLED IN IF YOU ARE ARRIVING AND DEPARTING BY AIRLINE

	Airline	Flight No.	Time
ARRIVAL	_____	_____	_____
DEPARTURE	_____	_____	_____

Enclosed is remittance in the sum of $_____ to cover _____ registrations at $10 each, if paid prior to November 4, or $13 each after November 4. Make all checks payable to 1620 Users Group.

Received of_____ $_____ for registration fee for Joint Western & Mid-Western 1620 Users Group Fall Meeting, November 9, 10, 11, 1964, University of Oklahoma, Norman, Oklahoma.

_____
(Officer, 1620 Users Group)

RETURN THIS FORM TO: Charles E. Maudlin
Computer Laboratory
University of Oklahoma
Norman, Oklahoma 73069

PROCEEDINGS OF THE

1620 USERS GROUP JOINT WESTERN-MIDWESTERN REGION MEETING

NOVEMBER 9-11, 1964

AT THE OKLAHOMA CENTER FOR CONTINUING EDUCATION
NORMAN, OKLAHOMA

In an effort to keep these Proceedings compendious, the Executive Board has requested that material normally found in the Program Library documentation, such as operating instructions, program listings, etc., not be reproduced in the Proceedings. Where interest extends beyond the content of the Proceedings, further information may be obtained from the author or the Program Library.

TABLE OF CONTENTS

PROGRAM AGENDA

Monday - November 9, 1964

8:00	Registration	East Concourse
8:30	Welcome --- Paul Bickford, Western Region President -- Forum Room IBM Announcements --- Dick Williams Use of DP Library --- Dave Dye	
10:00	Coffee	
10:30	Systems/360 NPL -- Jay Michtom (IBM)	Forum Room
12:00	Lunch	
1:00	Electrical Utility Team	Meeting Room A3

	Session I Forum Room	Session II Meeting Room A
1:00	Linear Programming -- Dr. S. T. Parker (3107)	Continuous Steel Beam Analysis & Design -- Stanley W. Woods (3072)
1:30	Linear Programming (Continued)	Automated Steel Building Design System -- L. L. Bathurst (1164)
2:00	Computerized Economic Evaluation -- T. I. Markland (IBM)	The Ohio State 1620 Programming System -- James Shaffer (3023)
3:00	Coffee	
3:30	New Linear Programming -- Harry Muller (IBM)	Administration of a Short Course in FORTRAN -- J. R. Wright (3320)
3:50	New L. P. (Continued)	An Analog-to-Digital Conversion System for the IBM 1620, Model I -- W. Wilcoxson (5045)
4:40	New L. P. (Continued)	PACTULAS - A. Digital Analog Simulator for the IBM 1620 - R. O. Brennan (IBM)
5:00	Adjournment of Day's Sessions	
7:30 p.m.	New Users Session ---	Paul Bickford -- Forum Room
8:00 p.m.	Sound-Off Session ---	C. E. Maudlin -- Forum Room

Tuesday - November 10, 1964

7:30 Late Registration — East Concourse

	Session I Forum Room	Session II Meeting Room A
8:00	Advanced FORTRAN II Techniques -- Lanny Hoffman (1177)	A Multivariate Analysis Package -- Rex L. Hurst (5103)
8:45	FORTRAN II (Continued)	Strap with FORMAT -- W. Wilcoxson (5045)
9:30	FORTRAN II (Continued)	Stochastic Simulation of Populations in Natural Selection Problems -- Alice M. Brues (3082)
10:00	Coffee	
10:30	FORTRAN II (Continued)	Application of Difference Equations in Numberical Analysis --John NcNamee (5171)
11:00	FORTRAN Flow Chart Program -- Robert E. Babcock (5058)	Difference Equations (Continued)
11:30	SUNDYNE Pump Quoting System -- Robert E. Babcock (5053)	Critical Path Method of Planning, Scheduling and Control for Construction Operations -- Jim S. Thompson
12:00	Luncheon	

Keynote Address:

A Computer's Reminiscent Remarks - Dr. Archie M. Kahan, Ph. D
Executive Director of the
University of Oklahoma
Research Institute

1:30	SPS Tutorial and Workshop --	Richard Pratt - Meeting Room A2
1:30	The Use of 1620-1311 System in Blood Bank Inventory Control -- Stanley Shannon (5226)	1620 FORTRAN IV - James White(3086)
1:45	Numberical Control - Tom Woods	FORTRAN IV (Continued)
2:15	Louisiana Tech Information Retrieval System -- David A. Hake (3087)	Programing for Selective Assembly -- John F. Mahan (3070)
2:45	Information Retrieval (Continued)	Experiences with MISS LESS and MISS LESS DATING SYSTEM - R. E. Dillion (3327)
3:00	Coffee	

Tuesday - November 10, 1964 (Cont'd)

3:30	STUFF - 1620 Universal Function Fitter -- IBM	Medical Panel: Chairman: Dr. Edward N. Brandt, Jr., M.D., Ph.D. O.U. Medical Research Computer Center Oklahoma City, Okla. Members: Dr. John Gustafson Iowa Methodist Hospital Des Moines, Iowa William F. Blose Baylor University College of Medicine, Houston, Texas Lawrence Newton M. D. Anderson Hospital Houston, Texas
5:00	Adjournment of Day's Session	
7:00 p.m.	Series of Four Movies: (1) Plotter, (2) Drafting, (3) Telstar, (4) Sketch-Pad	
9:00	Executive Board Meeting	

Wednesday - November 11, 1964

	Session I Meeting Room A	Session II Forum Room
8:00	Simulation the Businessman's Laboratory -- Bill Bryan (IBM)	Sort/Merge -- Vern Boyer (IBM)
10:00	Coffee	
10:30	Writing FORTRAN Subroutines in SPS for the 1620 Computer -- Roy J. Wepman (IBM)	Education Panel: Chairman: Dr. Richard V. Andree University of Oklahoma Mathematics Dept. Members: Woodfin C. Garrett High School Teacher Heavenear, Oklahoma J. R. Oliver, Dean Graduate School Director, Computing Center University of Southwestern Louisiana

Wednesday - November 11, 1964 (Cont'd)

Members: Jesse Richardson
Senior Supervisor in
Education
Department of Education
Boston, Massachusetts

Ronald Turner, Director
of Data Processing
Lansing, Michigan
Public Schools

11:30 FORTRAN II-D Subroutines
-- Frank Mozina (IBM)

12:00 Lunch

1:00 Frequently Used Computer Techniques That Do Not Work -- Richard V. Andree (5771)

Systems/360 Operation System -- John Giffin (IBM)

2:00 Use of 1620 for Typesetting
-- Bill Williams

2:45 IBM Reply to Sound-Off

Meeting Adjournment

ADAMS STERLING	TENN. STATE UNIV.	NASHVILLE TENNESSEE
ADKINS ALONZO F.	TEXAS TECH. COLLEGE	LUBBOCK TEXAS
AIGNER DENNIS J.	UNIV. OF ILL. COMMERCE C	ILLINOIS
AIKALA ROBERT G.	NORTHERN MICH. UNIV.	MARQUETTE MICHIGAN
ALBERTSON L. C. JR.	TECH. COMP. CENTER	SAN FRANCISCO, CALIF.
ALLAN R. M.	IBM	GLENDALE, CALIF.
ARMSTRONG DON	UNIV. OF MISSOURI	COLOMBIA, MISSOURI
ARMSTRONG NORMAN	HENN., DUR. AND RIC.	OMAHA, NEB.
ASHBAUCHER DAVE	COMPUTER LAB OU	NORMAN OKLA
BABCOCK BOB	SUNDSTRAND AVIATION	DENVER, COLORADO
BAKER EUGENE	HALLIBURTON CO.	DUNCAN OKLAHOMA
BARBER MILTON	LOMA LINDA UNIV.	LOMA LINDA, CALIF.
BATHURST L. L.	RUST ENGINEERING CO	BIRMINGHAM ALABAMA
BELONOS S. P.	COLUMBIA GAS SYSTEM	COLUMBUS 12, OHIO
BICKFORD P. A.	O.U. MED CENTER	OKLAHOMA CITY, OKLA.
BINGHAM DAVID	MIDWEST RESEARCH INS	KANSAS CITY MO
BLOSE WILLIAM F	BAYLOR UNIV OF MED	HOUSTON TEXAS
BOYER VERNON T.	IBM	SAN JOSE, CALIF.
BRADBURY H. M.	COMP. SYSTEMS ENGR.	NORMAN, OKLA.
BRADLEY ROBERT I.	UNIVERSITY TULSA	TULSA OKLA
BRENNAN ROBERT D	IBM RESEARCH LAB	SAN JOSE CALIF
BRIGHAM THOMAS	ILL. ST. UNIVERSITY	NORMAL ILLINOIS
BROOKS WENDELL	OHIO UNIVERSITY	ATHENS OHIO
BROWN JEAN M.	UNIVERSITY OF NEB.	LINCOLN, NEB.
BROWN ALICE	MCDONNELL AIRCRAFT	ST. LOUIS 66 MISSOURI
BRYANT CHARLES W.	US NAVEL AMM. DEPT	CRANE INDIANA
BUCHANAN ROBERT L.	U.S. BUR. OF RECLAM.	SACRAMENTO, CALIF.
BURGER B.	OMNIMETRICS	LOS ANGELES, CALIF.
BURROWS WILLIAM A.	DRAVO CORP	PITTSBURGH
BURZIO AGOSTINO	UNIV OF SANTA CLARA	SANTA CLARA CALIF
BYMARK JOHN V.	ERIE MINING	HOYT LAKES,MINN.
BYRNE MAURICE E.	IDAHO POWER CO.	BOISE, IDAHO
CAHALAN DONALD L.	BLACK & VEATCH	KANSAS CITY MISSOURI
CHASTAIN SANDRA	PITTS. PLATE GLASS CO.	CORPUS CHRISTI TEXAS
CHICK C. E.	GULF STATES UTILI	BEAUMONT TEXAS
CHURCH STEVE	OG AND E	OKLA CITY OKLA
CLARK STANLEY A.	P.B. SERVICE CO. OF N.H.	MANCHESTER N. HAMP.
CLAUGHTON J C	SW PB SERVICE CO	AMARILLO TEXAS
CLAY WILLIAM J	US ARMY ENGR DIST	WALLA WALLA WASH
CONFREY EVAN	HARTFORD STATE TECH	HARTFORD CONN
CONVERY LAWRENCE	CITY OF DETROIT	DETROIT, MICHIGAN
COPLAND GEORGE V	HALLIBURTON CO.	DUNCAN OKLA
CORNISH JOHN H	WEA. BUR. TULSA R F	TULSA OKLA
COTTON BILL	MET. STRUC. CORP.	GRAPEVINE, TEXAS
COX ED	MEMPHIS LIGHT GAS	MEMPHIS TENN.
DALLWIG KEN	CLARK OIL & REF. CO.	BLUE ISLAND, ILLINOIS
DANON REBECCA	WESTERN RESERVE UNIV	CLEVELAND OHIO
DAVISON DAVE	PUBLIC SERVICE CO	TULSA OKLAHOMA
DAVIS D. H.	TRUNKLINE GAS CO	HOUSTON TEXAS
DAVIDSON CH	UNIV WISCONSIN	MADISON WISC
DE LEGALL WALTER A	N Y MEDICAL COLLEGE	NEW YORK N Y
DECK JAMES C	INLAND STEEL COMPANY	HAMMOND INDIANA
DENISON GEORGE B	FAIRBANKS MORSE CO.	BELOIT WISC
DEVENNEY WILLIAM S	WABASH COLLEGE	CRAWFORDSVILLE IND
DIETZLER F. K.	GOODYEAR TIRE & RUBB	AKRON OHIO
DILLON R. E.	TRUNKLINE GAS CO	HOUSTON TEXAS

DOIG MARILYN	COLORADO STATE UNIV	FORT COLLINS
DONALDSON JACK S.	LEAR SIEGLER INC.	GRAND RAPIDS MICHIGAN
DRAVES RALPH H.	NW STATES PORTLAND	MASON CITY IOWA
DUFF ROBERT C	LEAR SIEGLER INC	GRAND RAPIDS MICHIGAN
DYE DAVID R	IBM	WHITE PLAINS NEW YORK
EDER JOE	COMPUTER LAB OU	NORMAN OKLA
EDGE ERVIN	COMPUTER LAB OU	NORMAN OKLA
ELLISON GEORGE	SANDIA CORP	ALBUQUERQUE N MEX
ELWELL WALTER G.	NEBRESKA WESLEYAN	LINCOLN NEBRASKA
ENYEDY GUSTAV	DIAMOND ALKALI CO.	PAINESVILLE OHIO
FARROW THOMAS H.	TAMPA ELECTRIC CO.	TAMPA FLORIDA
FINNEGAN KATHERINE	REMINGTON ARMS CO.	BRIDGE PORT CONN.
FREDERICKSON J. H.	BENHAM-BLAIR&AFFIL.	OKLAHOMA CITY OKLAHOMA
FULLER JERRY D	BAYLOR UNIV	WACO TEXAS
GAITHER JAMES D	COASTAL ST GAS PROD	CORPUS CHRISTI TEXAS
GARB FORREST A.	H.J.GRUY & ASSOCI.	DALLAS
GIBBONS I L	ARKANSAS POWER LIGHT	PINE BLUFF ARK
GILLOCK K E	SILAS MASON CO,INC	AMARILLO TEXAS
GONZALES RICHARD	BRADLEY UNIV	PEORIA ILLINOIS
GOODE JOHN M	HALLIBURTON CO	DUNCAN OKLA
GRAMLICH PAUL F.	LINE MATERIAL IND	ZANESVILLE OHIO
GREEN B E	ARKANSAS POWER LIGHT	PINE BLUFF ARK
GRIFFITH WALLACE	CENTRAL MISS. STATE	WARRENSBURG MISS.
GRIMES ROBERT	CENTRAL MISS. STATE	WARRENSBURG MISS
GROVE WENDELL E.	GENERAL MOTORS INST.	FLINT MICHIGAN
GRUNDY JANE P	UNITED STATES STEEL	MONROEVILLE PENN.
HAKE DAVID A	LOUISIANA TECH COMP	RUSTON, LOUISIANA
HALL O F	PURDUE UNIVERSITY	LAFAYETTE IND
HALL CAROL A.	UNIV. OF SW LA.	LAFAYETTE LOUISIANA
HALL JOHN L	MESTA MACHINE CO	PITTSBURGH PA
HAMMELMAN EDGAR D.	US NAVEL AMM DEPT	CRANE INDIANA
HARALAMPU GEORGE S	NEW ENGLAND ELEC SYS	BOSTON MASS
HARIG RICHARD F	DRAVO CORP	PITTSBURGH PENN
HARRIS RICHARD A.	NORTH TEX. ST. UNIV.	DENTON, TEXAS
HARRISON KENNETH L	SOUTHER NAT. GAS CO.	BIRMINGHAM ALA
HART WALTER	TIDEWATER OIL CO.	OKLAHOMA CITY, OKLA.
HATFIELD FRED A.	LINE MATERIAL IND	ZANESVILLE OHIO
HELYER RALPH	SW PB SERVICE CO	AMARILLO TEXAS
HESS J. I.	THIOKOL CHEMICAL CO.	MARSHALL TEXAS
HETLAND LEROY D	UNIV. OF SANTA CLARA	SANTA CLARA CALIF
HEWETT SYLVIA	DU PONT DE NEMOURS	FLINT MICHIGAN
HINTZ ANN	IBM	CHICAGO, ILLINOIS
HITAMI FAROUK M EL	E WASH STATE COLLEGE	CHEVEY WASH.
HOFFMAN VICTOR W.	US WEATHER BUREAU	FORT WORTH, TEXAS
HOFFMAN LANNY	PRINCETON UNIV	PRINCETON NEW JERSEY
HOKE TOM	OKLA GAS & ELECTRIC	OKLA CITY OKLA
HOLLOWAY W D	TRUNKLINE GAS CO.	HOUSTON TEXAS
HOLMES JACK	COOPER BESSEMER CORP	MOUNT VERNON OHIO
HORTON DON W	IBM	WHITE PLAINS NY
HORTON MURIEL	JET PROPULSION LAB	PASADENA, CALIF.
HUHTA J. MICHAEL	IBM CORP.	DENVER, COLORADO
HURST REX L	UTAH ST UNIV	LOGAN UTAH
JARDINE D. A.	DU PONT OF CANADA	KINGSTON ONTARIO
JENKINS ORVILL	H J GRUY ASS.	DALLAS TEXAS
JOHNSON ALAN D.	NATIONAL AERO & APAC	SANDUSKY OHIO
JONES WAYNE	ADF INDUSTRIES INC.	ALBUQUERQUE, N.M.

JONES CLINTON E.	TENN A&I STATE	NASHVILLE
JONES IVAN J	CORPS OF ENGRS ARMY	PORTLAND OREGON
JONES W M	ARKANSAS POWER LIGHT	PINE BLUFF ARK
JUSTICE R	GENERAL MOTORS	MILFORD MICHIGAN
KATZ ELI	DEPT OF WATER & POW	LOS ANGELES CALIF
KEELING ROY V.	DUGWAY PROVING GROUND	DUGWAY UTAH
KERR H. B.	TENNESEE TECH	COOKE VILLE TENN.
KERRIGAN FELIX J	V A HOSPITAL	OMAHA NEBR.
KILLIAN DONALD R.	CENTURY ELECTRIC CO.	ST. LOUIS MISSOURI
KLEIN MICHAEL	UNIVERSITY OF AKRON	AKRON OHIO
LAMOREUX WALLACE W	WEATHER BUREAU HYDRO	WASHINGTON D. C.
LANG WILLIAM D.	FLORIDA POWER & LIGH	MIAMI FLORIDA
LARCADE GEORGE A	HALLIBURTON CO	DUNCAN OKLA
LAWRENCE DEAN	MIDWEST RESEARCH INS	KANSAS CITY MO
LEESON DN	IBM	WHITE PLAINS NY
LERICK GEORGE	ERIE MINING COMPANY	HOYT LAKES, MINN.
LIPSON ALVIN L.	VIRGINIA ELEC. & PO	RICHMOND VIRGINIA
LITTELL ARTHUR S	WESTERN RESERVE UNIV	CLEVELAND, OHIO
LOCKE GERALD W.	TEXAS TECH. COLLEGE	LUBBOCK TEXAS
LOGAN S. WM.	P. R. MALLORY & CO	INDIANAPOLIS INDIANA
LOUIS JENE Y.	LONG ISLAND LIGHTING	HICKVILLE NEW YORK
MACMULLIN BRUCE R	WESTERN SUPPLY CO	TULSA OKLA
MAGEE ROLAND H.	THE MAGNAVOX CO.	FT. WAYNE
MAHAN JOHN F	GEN. MOTORS INST.	FLINT MICH.
MANNING BOB N	GOODYEAR AEROSPACE	LITCHFIELD PARK ARIZ
MARKS MAXWELL	IBM CORP.	WHITE PLAINS, N.Y.
MARKLAND THOMAS I.	IBM	POUGHKEEPSIE, N.Y.
MARTIN W B	OKLA GAS & ELECTRIC	OKLA CITY OKLA
MARTIN DONALD J	US PUBLIC HEALTH SEV	LAS VEGAS NEV
MASKIELL FRANK H.	PENNSYLVANIA TRANSFORMER	BOX 330 CNAONSB.PENN.
MAUDLIN CHARLES	COMPUTER LAB OU	NORMAN OKLA
MCATEER MARY LYNN	JONES & LAUGHLIN	PITTSBURGH PENN.
MCCALL ALLEN W.	H.J.GRUY & ASSOCI.	DALLAS TEXAS
MCCOLLUM PAUL A.	OKLA. STATE UNIV.	STILLWATER, OKLA.
MCDONOUGH J A	TEXAS GULF SULPHURCO	NEWGULF TEXAS
MCDONALD D. R.	FED RES BANK	MINNEAPOLIS MINNES.
MCKEE LOWRY L.	UNIV. OF N. CAR.	RALEIGH, N.C.
MCMULLIN BEN	SW PB SERVICE CO	AMARILLO TEXAS
MEAGHER JACK R.	WEST. MICH. UNIV.	KALAMAZOO, MICH.
MEHL JAMES	COMPUTER LAB OU	NORMAN OKLA
MERGEN F C	BRADLEY UNIV	PEORIA ILLINOIS
MILLER WALTER E.	PROCTER AND GAMBLE	CINCINNATI, OHIO
MOORE PAUL A	HALLIBURTON CO.	DUNCAN OKLA
MORRISON JACK L.	OIL INFORM. CENTER	NORMAN, OKLA.
MORTON E. L. JR	LOUISIANA STATE UNIV	BATON ROUGE LOUSIANA
MOWCHAN MICHAEL S.	MONSANTO CO.	MIAMISBURG OHIO
MOZINA FRANK	IBM CORP	PITTSBURGH PA
MULLER HARRY W	IBM	WHITE PLAINS N Y
NEWTON LAWRENCE E JR	MD ANDERSON HOSP	HOUSTON TEXAS
NICHOLS DICK	IBM	LOS ANGELES, CALIF.
NORRIS BOYD C	BUR. OF RECLAMATION	SACRAMENTO CALIF
OESTERLEI ROBERT E.	CENTURY ELECTRIC CO.	ST. LOUIS MISSOURI
OLDFATHER FELICIA	PIONEER HI-BUD CORN	DES MOINES
OLIVER JAMES R.	UNIV. OF SW LA.	LAFAYETTE LOUISIANA
OLOUGHLIN JOHN B	FORT HAYS ST COL	HAYS KANSAS
OLSON EARLE E.	NEBRASKA CONSOL. MILLS	OMAHA NEBRASKA

ORLOFF MILTON J.	GENERAL MOTORS INST.	FLINT MICHIGAN
ORNDORFF G. ROBERT	U.S.PUBLIC HEALTH	ROCKVILLE MARYLAND
PANA MILTON T.	KENNE. COPPER COPR.	SALT LAKE CITY, UTAH
PAQUIN NANCY	U.S.PUBLIC HEALTH	ROCKVILLE MARYLAND
PARENT L. V.	TRUNKLINE GAS CO	HOUSTON
PARKER C	GENERAL MOTORS	MILFORD MICHIGAN
PARKER THOMAS S	KANSAS ST UNIV	MANHATTAN KANSAS
PENDERGRASS J. D.	S.W. POWER ADMIN.	TULSA, OKLA.
PETERSON DONALD E	N. DAKOTA STATE	FARGO NO. DAKOTA
PIERCE MELVIN L.	ARLINGTON STATE COLL	ARLINGTON PUERCE
POORE JESSE H.	LOUISIANA TECH COMP	RUSTON, LOUISIANA
POPE WENDELL L	UTAH ST UNIV	LOGAN UTAH
PORTER CHARLES	ILL ST UNIV	NORMAL ILLINOIS
POWELL JOHN E.	UNIV.OF SOUTH DAKOTA	VERMILLION S DAKOTA
PRATT RICHARD L.	DATA CORP.	DAYTON, OHIO
PRICE ROBERT TRACEY	N.A.S.A.	HUNTSVILLE, ALA.
QUO PHILIP	BLACK & VEATCH	KANSAS CITY MISSOURI
RADLE W J	AIRTEMP DIV CHRYS CO	DAYTON OHIO
RANTZ MORRIS	SW PB SERVICE CO	AMARILLO TEXAS
RAYMOND SHIRLEY A.	OHIO AGRI. EXPERI.	WOOSTER OHIO
REDMOND JOHN L	SOUTHERN SERVICES	BIRMINGHAM ALA
RENICK HARRY	WEYERHAEUSER CO.	TACOMA WASH
RICHARDSON JAY	ILL ST UNIV	NORMAL ILLINOIS
RICHARD CHARLES W	AIR FOR INST OF TECH	W-PATTERSON AFB OHIO
RIGGIN E. E.	S. W. POWER ADMIN.	TULSA, OKLA.
ROBERTSON LEWIS	HALLUBURTON	DUNCAN OKLA
RUBENSTEIN M.	OMNIMETRICS	LOS ANGELES, CALIF.
SALOMON MARRIAN	SANDIA CORP.	ALBUQUERQUE NEW MEXICO
SANDERS PAUL G	ABBOTT LAB	NORTH CHICAGO ILL
SEACAT R. H. DR.	TEXAS TECH. COLLEGE	LUBBOCK TEXAS
SECOR KENNETH E	CHICO ST COLLEGE	CHICO CALIF
SHAFFER JAMES	THE OHIO STATE UNIV.	COLUMBUS OHIO
SHANNON B STANLEY	WADLEY RESEARCH INST	DALLAS TEXAS
SHARP HUGH H III	US ARMY CORPS OF ENG	ALBUQUERQUE NEW MEX
SINGER SEYMOUR	SAN FRANCISCO STA E	SAN FRANCISCO CALIF
SMITH R. E.	GULF STATES UTILI	BEAUMONT TEXAS
SMITH DEAN C.	FELS RESEARCH INST.	YELLOW SPRINGS OHIO
SMITH KENNETH	CABOT CORPORATION	PAMPA TEXAS
SMITH NOEL T	IND. STATE COLLEGE	TERRE HAUTE IND
SMITH KENNETH	CABOT CORPORATION	PAMPA TEXAS
SNAVELY CAROL J	OPTICAL COATING LAB	SANTA ROSA CALIF
SOCKS BRUCE J.	IBM	CHICAGO, ILLINOIS
SPRADLING GARY	COMPUTER LAB OU	NORMAN OKLA
STEELE LAURA B.	GENERAL MOTORS INST.	FLINT MICHIGAN
STEINHART R. F.	IBM CORP.	NEW YORK
STRINDBERG WILLARD R	CHANDLER EVANS CORP	WEST HARTFORD CONN
STROUSE E. I.	COLUMBIA GAS SYSTEM	COLUMBUS OHIO
SWINDLE G	OMNIMETRICS	LOS ANGELES CALIF
TAYLOR GEORGE I JR	LA FLOOD CONTROL DIS	LOS ANGELES CALI
TAYLOR DAN L	S.DAK.SCH.OF MINES.TECH	RAPID CITY S DAK
TEMPLE JOHN M.	UNIV OF CHATTANOOGA&TECH	CHATTANOOGA TENN.
THAYER RAYMOND J.	LINE MATERIAL IND.	MILWAUKEE
THOMPSON MYRL W.	OLDSMOBILE DIVISION	LANSING 21 MICHIGAN
THOMAS R. B.	FED RES BANK	MINNEAPOLIS MINNESOTA
THOMAS ROBERT J.DR.	DE PAUW UNIV.	GREENCASTLE INDIANA
THOMPSON W. SAM	HALLIBURTON CO.	DUNCAN OKLAHOMA

TUCKER ROY W	HUMBOLDT ST COLLEGE	ARCATA CALIF
VOLLMAR NAOMI	FLINT COMMUNITY COLLEGE	FLINT MICHIGAN
WALDEN JACK M.	OKLA. STATE UNIV.	STILLWATER, OKLA.
WALDON JERRY H.	NORTH TEX. ST. UNIV.	DENTON, TEXAS
WALDEN WILLIAM E.	UNIV. OF OMAHA	OMAHA NEBRASKA
WALKER JAMES T.	COAST AND GEOD. SUR.	EL PASO, TEXAS
WALKER MARY C.	WHI. SAN. MISS. RAN.	WHITE SANDS, N.M.
WANTUCK WILLIAM H	SOUTHER NAT. GAS CO	HOUSTON TEXAS
WARREN CARL	OKLA GAS AND ELEC	OKLA CITY OKLA
WELLS GARLAND R.	LOUISIANA TECH COMP	RUSTON, LOUISIANA
WELLS FRANK J.	LONG ISLAND LIGHTING	HICKVILLE NEW YORK
WEPMAN R J	IBM	POUGHKEEPSIE N Y
WHITE ROBERT R.	L.A. DEPT WATER & POWER	LOS ANGELES CALIF.
WIGDAHL ALLEN B.	ALLEN-BRADLEY CO	MILWAUKEE WISCON.
WILCOXSON W. L.	N. CIVIL ENGR. LAB	PORT HUENEME, CALIF.
WILLIAMS DICK	IBM	LOS ANGELES, CALIF.
WOMBLE L R	SILAS MASON CO,INC	AMARILLO TEXAS
WOODS STANLEY W.	WIC. HIGHWAY COMMI.	MADISON
WOODS TOM	IBM CORP	OKLA CITY OKLA
WORRELL R B	SANDIA CORP	ALBUQUERQUE N MEX
WRIGHT JAMES R	THE TRANE CO	LA CROSSE WISCONSIN
YATES TOM	AERO COMMANDER	NORMAN OKLA

Linear Programming

(S. T. Parker Kansas State Univ.)

Linear Programming involves the finding of an arrangement of a number of resources which are subject to some linear constraints, which arrangement will make a certain linear expression a maximum (or minimum, as desired in the problem). The problems of maximizing profit, output, etc. or of minimizing costs, time, distance, etc. can frequently be solved by linear programming methods.

The linear constraints are of three kinds, as indicated by the examples:

$$s + 3t \leqq 10$$

$$x_1 + x_2 - 6x_3 + 5x_4 = 2.68$$

$$2x_1 + 5y - 3z \geqq 7$$

The method described in the following sections is one version of the so-called "simplex" method. The variables specified in the constraints, and in the linear expression to be maximized (or minimized), will be called "real" variables. In addition we will introduce "slack" and "artificial" variables. In all our work no variable real, slack, or artificial -- will assume a negative value.

Our discussion will center around the maximizing problem, with a brief final note to show how the minimizing problem can be converted so as to have once more a maximizing situation. We will assume that the problem is "feasible". Thus, we will not have a set of contradictory constraints such as

$$x \leqq 5, \quad y + z \leqq 7$$

$$x + 2y + 2z \geqq 25$$

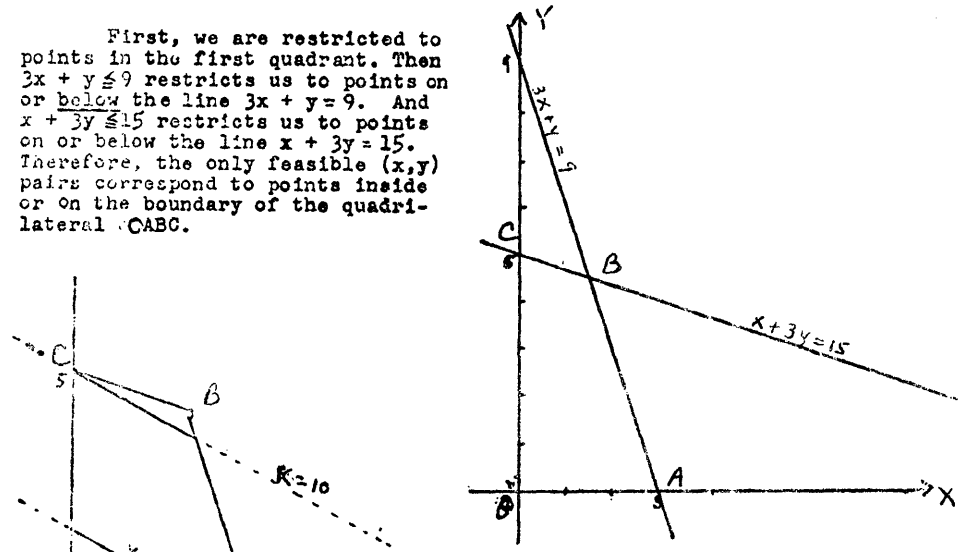
There is an extensive literature dealing with linear programming methods. This note gives a very brief introduction to the subject; references for further reading are readily available.

Example 1. Maximize $x + 2y$, given

$$3x + y \leqq 9, \quad x + 3y \leqq 15$$

(Remember that $x \geqq 0$, $y \geqq 0$).
We shall solve this problem graphically and then show how the simplex method is applied to yield the same result.

First, we are restricted to points in the first quadrant. Then $3x + y \leqq 9$ restricts us to points on or below the line $3x + y = 9$. And $x + 3y \leqq 15$ restricts us to points on or below the line $x + 3y = 15$. Therefore, the only feasible (x,y) pairs correspond to points inside or on the boundary of the quadrilateral OABC.



What to do about the $x+2y$ expression? Let us draw a series of straight lines $x+2y = k$, with k taking on a number of values, as shown in the figure. It is clear that the farther the line is from the origin the larger the value of k. It is also clear that the member of the family of parallel lines which is farthest from the origin and still has contact with the quadrilateral will pass through B. Therefore, the x and y, corresponding to the point B, will maximize $x+2y$, subject to the constraints given.

Generalized. In the above case our region of feasibility consists of the interior and boundary of a quadrilateral -- a convex quadrilateral. From the linearity of the constraints involved, a set of linear constraints in two dimensions will lead to a convex polygon.

When we have three dimensional situations, the linear relations become planes, and the region of feasibility then becomes a convex polyhedron. The maximizing of the linear expression involves finding that one of a family of parallel planes which is farthest from the origin.

For more than three dimensions, we have convex sets of higher order. The theory of convex sets is basic to the theory of linear programming, but we shall discuss only a few of the concepts and dispense with proofs.

In omitting proofs we shall appeal to the intuition. Note that, for a linear expression, $z = c_1x_1 + c_2x_2 + \cdots + c_nx_n$, the partial derivatives of z are non-zero. This means that no absolute maximum or minimum can exist inside the feasibility region. Thus, maximum and minimum must occur on the boundary and, in particular, at a corner.

Simplex Method. In the simplex method, we begin our considerations by setting all the real variables zero. Let us rewrite the inequalities above as equalities, by introducing "slack" variables:

$$3x + y + u = 9$$

$$x + 3y + v = 15$$

$$(u \geq o,\ v \geq o).$$

Thus, initially, x = o, y = o, u = 9, v = 15.

Let us check the four "corners" of the quadrilateral drawn above.

At O: x = o, y = o, u = 9, v = 15

At A: x = 3, y = 6, u = o, v = 12

At B: x = 3/2, y = 9/2, u = o, v = o

At C: x = o, y = 5, u = 4, v = o

Note that, at each corner, two variables only have non-zero values. The variables which have the non-zero values at a corner are said to form "the basis" (at that corner). In the simplex method, when we have M constraints, we always have exactly M basis variables.

Let us re-write the equalities:

$$x_1 + 3x_3 + x_4 = 9$$

$$x_2 + x_3 + 3x_4 = 15$$

and try to maximize $z = ox_1 + ox_2 + 1x_3 + 2x_4$.

We set up a tableau as follows:

Basis	c	1	2	3	4	
	c	-1	0	1	2	0
1	0	1	0	3	1	9
2	0	0	1	1	3	15
		0	0	-1	-2	0
		a	b	d	e	f

a = -1.0 + 0.1 + 0.0
b = -1.0 + 0.0 + 0.1
d = -1.1 + 0.3 + 0.1
e = -1.2 + 0.1 + 0.3
f = -1.0 + 0.9 + 0.15

The number f is the value of z for the particular corner under consideration. Initially, as given here, with basis (1,2), variables $x_1 = 9$, $x_2 = 15$, $x_3(=x) = 0$, $x_4(=y) = 0$, and $z = f = 0$.

To improve the situation we check the last row of the tableau for negatives. Whenever a negative appears, the value of z can be improved by introducing some of the corresponding variable into consideration. The greatest negative is -2, corresponding to variable x_4. Let us check to find the greatest value x_4 can assume. We divide any positive coefficients in the x_4 column into the right hand side. (the constraints column). We see that max. $x_4 = 9$ from the first, and max. $x_4 = 5$ from the second. Therefore, max. $x_4 = 5$.

If x_4 is going to be 5, not zero, then it must go into the basis. It will replace x_2, since $x_4 = 5$ is obtained by letting x_4 have its maximum value in the 2nd equation. In order that we be able to read off from the right side immediately, the value of x_4 (in the basis), we divide the coefficients in the second equation by 3, the coefficient of x_4. We have the arrangement shown in the tableau on the left. It is necessary to eliminate all other coefficients in the x_4 column. This is as indicated on the right. Note that there is one negative in the last row.

Basis	c	1	2	3	4	
	c	-1	0	0	1	2	0
1	0	1	0	3	1	9
4	2	0	1/3	1/3	1	5

Basis	c	1	2	3	4	
	c	-1	0	0	1	2	0
1	0	1	-1/3	8/3	0	4
4	2	0	1/3	1/3	1	5
		0	-2/3	-1/3	0	10

At this stage, pause to note that $x_1 = 4$, $x_4 = 5$, with $x_2 = 0$ and $x_3 = 0$. This corresponds to the point C. Thus, we have progressed from the corner O to the corner C and Z has been advanced from 0 to 10.

One more iteration produces the result shown, giving $x_3 = 3/2$, $x_4 = 9/2$, $x_1 = 0$, $x_2 = 0$,

which corresponds to the point B. Note that the maximum Z is 21/2.

Basis	c	1	2	3	4	
	c	-1	0	0	1	2	0
3	1	3/8	-1/8	1	0	3/2
4	2	-5/8	3/8	0	1	9/2
		1/8	5/8	0	0	21/2

Artificials. What about the = and ≥ constraints? How can we start off with all real variables zero? We manage this by introducing an "artificial" variable, assign to it a large "penalty" cost, and try to eliminate it by the process as outlined above.

For the constraints

$$2x + 3y + z = 5,$$

and

$$2x + 3y + z \geq 5,$$

we write

$$u + 2x + 3y + z = 5$$

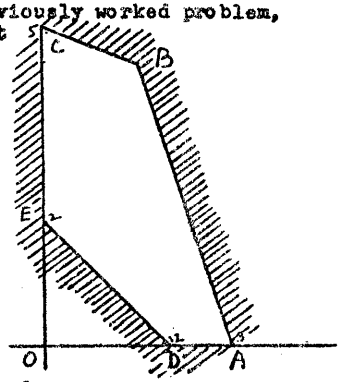
and

$$u + 2x + 3y + z - v = 5,$$

respectively. The "artificial" u is assigned a high penalty coefficient, while the slack v has, as before, a zero cost coefficient. In each case, since we have x = 0, y = 0, z = 0 initially, we have the initial value of u = 5.

To show how the artificial variable is introduced and then "priced out" in practice, consider the previously worked problem, to which is added the additional constraint

$$x + y \geq 2.$$



The graphical picture is as shown on the right. The feasibility region then is the interior and boundary of the (convex) polygon DABCE. When we begin with real variables zero, we are really at point O, outside the region entirely. That is where the "penalty" and "artificial" features enter.

Renumbering, as before, gives us the problem:

Maximize $0x_1 + 0x_2 - 1000x_3 + 1x_4 + 2x_5 + 0x_6$,

where $x_1 + 3x_4 + x_5 = 9$

$x_2 + x_4 + 3x_5 = 15$

$x_3 + x_4 + x_5 - x_6 = 2$

We have, arbitrarily, chosen the "high" penalty -1000 to be assigned to the artificial x_3.
Successively we have the tableaux:

point 0

b	c	1	2	3	4	(5)	6	
	-1	0	0	-1000	1	2	0	0
1	0	1			3	1		9
2	0		1		1	3		15
(3)	-1000			1	1	1	-1	2
		0	0	0	-1001	(-1002)	1000	-2000

9, 5, (2)

b	c	1	2	3	4	5	(6)	
	-1	0	0	-1000	1	2	0	0
1	0	1		-1	2	0	1	7
(2)	0		1	-3	-2	0	3	9
5	2			1	1	1	-1	2
		0	0	1002	1	0	(2)	4

point E

7, (3), X

point C

b	c	1	2	3	(4)	5	6	
	-1	0	0	-1000	1	2	0	0
(1)	0	1	-1/3	0	8/3	0	0	4
6	0		1/3	-1	-2/3	0	1	3
5	2		1/3	0	1/3	1	0	5
		0	2/3	1000	(-1/3)	0	0	10

(3/2), X, 15

point B

b	c	1	2	3	4	5	6	
	-1	0	0	-1000	1	2	0	0
4	1	3/8	-1/8	0	1	0	0	3/2
6	0	1/4	1/4	-1	0	0	1	4
5	2	-1/8	3/8	0	0	1	0	9/2
		1/8	5/8	1000	0	0	0	21/2

Note that the optimizing point, B, corresponds to

$$x_4 = 3/2,\ x_6 = 4,\ x_5 = 9/2,\ \text{all other } x\text{'s} = 0.$$

Minimizing Problem. If the problem is one of minimizing

$z = c_1 x_1 + c_2 x_2 + \cdots + c_k x_k$, one can reduce it to the preceding case by maximizing

$$z_1 = c_1' x_1 + c_2' x_2 + \cdots + c_k' x_k ,$$

where $c_i' = -c_i$. When we find the maximum of this Z_1 to be -K, the answer to the original problem is minimum Z = K. All other parts of the simplex proceed as before -- including the introduction of slacks and artificials, and beginning with all real variables = 0 initially. (Whether the c's are positive or negative, the artificials are assigned "large" negative cost coefficients. How large is a matter of choice -- 100 times the largest of the others is safe.)

"COMPUTERIZED" ECONOMIC EVALUATIONS

November 2, 1964

T. I. Markland
IBM DATA SYSTEMS DIVISION
Poughkeepsie, New York

1.0 INTRODUCTION

Most industrial, scientific or business projects require capital investment decisions which raise the questions: Should we rent or buy, buy or make, make or rent, etc.? Whenever there is more than one method of implementing a project, the least costly approach must be found, tempered with management considerations, and then applied.

Suppose a company had to decide whether it should sell a car or a fleet of cars after one, two, three, or four years of use. If this decision was to be based solely on a financial evaluation to the exclusion of management considerations, a computer could be used to solve the problem. This problem will be treated, after reviewing the general economic evaluation process of the Capital Investment Program (CIP III). This program introduces a uniform approach to solving capital investment problems, reduces the probability of mathematical error, and increases management certainty to a degree which previously was unattainable with manual calculations.

Some of the required attributes of a "computerized" economic evaluation system are:

- GENERALITY
  The requirement for having a system sufficiently general to cover almost any type of project, regardless of its investment or expense pattern.
- FLEXIBILITY
  The ability to vary input information within the system to allow for changes in tax rates, tax lives, and cost rates.
- ACCURACY
  The necessity for giving the correct results based on economic principles in a format which can be conveniently interpreted.

A good starting point in the development of the system is to define all cost categories (Exhibit I). These general categories are:

- FIXED INVESTMENT
  This includes all capital expenditures such as tooling, test equipment, etc.
- INVESTMENT EXPENSE
  This includes all one-time costs which support fixed investments.
- OPERATING EXPENSE
  This includes all recurring costs throughout the life of the program.
- CURRENT INVESTMENTS
  These are inventory stock and prepaid investments such as insurance, etc.
- EXPENSE OR GAIN SUBJECT TO TAX
  These include capital gains, income, spoilage and scrap, write-offs, etc.

2.0 PROCEDURE

When the costs have been identified, the procedure for computing a net cost or net cash outlay may be developed. Each cost has different economic characteristics with the exception of investment expense and operating expenses which are separately identified and individually totaled.

Fixed investment is by far the most difficult to calculate because of the depreciation tax lives for different types of capital equipment. These tax lives change periodically and are therefore entered as data rather than program constants. From these tax lives, strings of depreciation per cents are generated for each year of the tax life. This is the per cent of the original investment which would be written off the book value in each year (Exhibit II).

Assuming these per cents have been generated for each type of investment, each fixed investment, as it appears in the input, is multiplied by the proper per cent in the proper year. This generates a string of depreciation dollars for each investment. These strings are added by year, giving the total normal fixed investment write-offs used for calculating tax credits. Recent tax legislation however, allows for an extra incentive tax credit in the year of the investment. This is calculated by multiplying each investment by its incentive tax credit rate and summing the results.

An additional manipulation must be made where the project produces a product which will be rented, and for which the company chooses to defer its tax credits over the tax life (Exhibit III). In this case the previously generated string is re-depreciated (or re-cycled) producing a new string to be used for tax credit purposes.

Next, investment expense, which includes design, debug, prototype build and training costs, etc., are considered. These costs are added to the operating expenses. A feature which standardizes the analysis and somewhat relieves the accounting burden, is placing man/year dollar rates and benefit rates on control cards. Thus, the input for manpower costs consists only of time (in man years) and the dollar value is computed automatically.

Each operating expense sub-category such as direct labor costs is added to the general expense category along with the investment expenses. An expense sometimes overlooked in economic analysis is carrying cost. This expense is included by maintaining an inventory carrying cost and inventory turnover rate on the control cards. Costs of inventory, materials and operating supplies are added, divided by the turnover rate and the resulting quotient is multiplied by the carrying cost rate. The product is then added to the operating expenses.

Current investments, in general, do not follow a typical cash flow pattern, because inventory and prepaid items are project assets which are not necessarily depleted in the year of the expense. To calculate the cash which has been expended or the tax credit claimed, it is necessary to know the past value of the asset. Since a computer can store the past value, the input need only reflect the present value of the asset. If the value of the asset increases, there is an expense; if it decreases, there is a depletion which generates a tax credit.

The last general category is tax impact dollars. The sale of fixed assets is subtracted from the net cash outlay and compared to the write-offs to find the capital gain or loss. Any loss is added to the depreciation, spoilage, etc., to compute the tax credit; any gain is multiplied by the capital gain tax rate and the resulting product is added to the net cash outlay (Exhibit IV).

Income is subtracted from net cash outlay and compared with expenses to find the net income or expense. Income tax is computed on any income exceeding the expenses and then added to net cash outlay. Expenses exceeding income are added to the depreciation, etc., to compute tax credit.

Tax credit is computed by taking the sum of the depreciation, extra write-offs, expenses, etc., and multiplying by the combined federal and state income tax rates. This product is then subtracted from the net cash outlay.

It should be noted that the above income is not the income of the project. The evaluation is performed under the assumption that income from the project will be the same, regardless of the chosen alternative. However, one alternative might include an extra service which would bring in an income which another alternative would not. For example: In evaluating the purchase or rental of a piece of equipment, if the equipment is bought outright, there may be additional time which can be used to produce an income beyond the scope of the project. Revenues thus derived would be entered on the input sheet under income.

It was mentioned previously that projects which produce rented products defer tax credits. This has been reviewed under fixed investments but is applicable to other types of investments and expenses. The tax credit on these items must also be deferred over the tax life of the product.

To this point, the quantity which is the net cash outlay for one of the alternatives of implementing the project has been calculated. If the same calculation is performed for another alternative, a second net cash outlay is derived for comparison purposes. The difference between these two alternatives is the cash investment or saving differential. If this process is repeated for each year of the project, a string of cash investments or savings (one for each year) or a cash flow pattern is developed.

Next the computer program tests for a conventional cash flow pattern. If the test finds a non-conventional pattern, the possibility of a multiple root equation exists which will give two or more answers (Rate of Savings). This problem is eliminated by paying off deferred investments (those which occur after savings) with prior savings. Any saving derived when comparing the two alternates is money which was not removed from the reserve assets of the corporation to support the project. Investments must be removed from the reserve assets. The rate of interest on the savings portion of the reserve assets (cost of capital) is applied to the savings to offset the deferred investments. If all the deferred investments are "compensated for" or if they deplete the savings, a conventional investment or saving pattern results.

The payback period is the number of years, starting with the present year, required for the sum of the savings to exceed the sum of the investments.

The last item to be calculated is "rate of savings." Rate of savings is equivalent to the compound interest required to equate the discounted or present value of the investment to the discounted or present value of the savings. The program calculates positive rates of savings, between 0% and 100% when alternative B is better than alternative A. If alternative A is better than alternative B, the rate of savings will be typed as "less than zero."

The input form is a three-part snap-out. The first and third part are duplicates and the second part is the keypunch layout (Exhibits V and VI). The form has two sections, one for each alternative. The top section is for the A, or Present Alternative, and the bottom is for the B, or Proposed Alternative.

Since the program compares B against A the analyst should use section B for the apparently better of the two alternatives; if he gueses wrong they can be reversed. There must be one page for each year of the program except those years requiring no entries. The first entry on the card is PROJECT NO. There must be an entry in this block for each page of the evaluation. The next block, labeled YEAR, contains the year of the costs entered. Place the project name over DESCRIPTION. ECONOMIC LIFE may contain from one to twenty-five years (the useful life of the project). The DATE block must contain the present date.

Either COST COMPARISON or COST REDUCTION evaluation may be chosen. COST COMPARISON is chosen if the alternatives are for a new project, and COST REDUCTION is chosen if alternative B is a replacement for A, already in operation. The only difference in the program is that PAYBACK is not printed on a COST COMPARISON. The RECYCLE entry determines which depreciation routine is used.

Each alternate is divided into the five previously described major expense categories. Each detail entry is in dollars and may contain up to seven digits. The categories for manpower are in hundredths of a man/year.

3.0 EXAMPLE

NOTE

The following values are fictitious and in no way reflect IBM policy or state or federal tax values.

One of the program outputs is a listing of the constants (Exhibit VII). Note the constants for the automobile problem: 4.00 years as the Fixed Investment Tax Life for Other; 2.00 years for the Inventory Turnover Rate for both alternatives; 1.7 per cent for Carrying Cost; 50.0 per cent for Federal Tax Rate; 5.0 per cent for State Tax Rate; 25.0 per cent for Capital Gain Tax Rate; and, 1.5 per cent for the Cost of Capital.

The constants are maintained on two cards which are read-in before the first job. They are re-read only if the constants are changed between runs (batching).

The first evaluation of the automobile problem is a comparison of selling the car after one and two years of use (Exhibit VIII). In alternate A the car will be kept for one year and in alternate B the car will be kept for two years. In this example, Economic Life will be two years.

The first concern is the cost of the car which in this example is $1,945. This figure is entered for both alternatives under Fixed Investment Other. The next consideration is Insurance which also will be the same for both alternatives in the first year. The $200 cost of insurance is not entered under Current Investment Prepaid Insurance because it is a one-year policy, but rather under Annual Cash Operating Expense - Other.

A material stock cost of $62 for oil, grease, oil filters, and air cleaners has been assumed for each year. The $7 rework cost is for brake and headlight adjustment. The $26 for tire rotation and engine tune up will not be added if the car will be sold in the next year.

At the end of the first year the car will be sold in alternative A. The wholesale value of this car one year old is $1,330, which is entered under Sale of Fixed Assets. The program automatically calculates the depreciation on the car for this year. When the car is sold, all of the assets must be removed from the company books, which means an additional write-off of 60% of the cost of $1,167.

In the second year (Exhibit IX) alternative A has the same entries as the first year except for the additional depreciation which the program automatically calculates. The program writes off all fixed investments in the last year of the porject. Since the cars were written off when they were sold, the normal depreciation must be offset by the total write-off (already claimed) of $2,334. This may be done by placing this value in the last (unlabeled) block under Annual Expense or Gain Subject to Tax.

For the two-year-old car, insurance cost is reduced by $10, the car is sold for $1,015, and the undepreciated amount of $584 is written off.

Examination of the output (Exhibit X) shows that the fixed investment for alternate A is double that of B, because in A we bought two new cars. The operating expense is slightly higher ($16) for alternative B, and current investments and investment expense are zero for both alternatives.

The total cost for alternative A is $4,430 as opposed to $2,501 for B. These figures include only cost and do not include tax credits.

In the CASH FLOW column for the first year, alternative A costs $1,300 less than B. This requires an outlay of $1,300 for alternative B which is in excess of the cost for alternative A. The second year shows a flowback of $1,461 for alternative B, giving a total net flowback of $161 of B over A. How does this compare to the total costs? The obvious answer is that the savings are less than expected when considering only the costs without considering the credits.

The values in the INV or SAV column are the same as those in the CASH FLOW column because the cash flow is conventional and therefore the COST OF CAPITAL has no bearing on the result of 12.4 per cent.

The 12.4 per cent represents the rate of return of alternative B over alternative A. Since there is never any discounting in the present year, the discounted value is the same as the INV or SAV value. The INV or SAV from the second year has been discounted until its discounted value is as close as possible to the investment in the previous year. In other words, if we place $1,300 in an investment account giving an interest rate of 12.4 per cent, one year later we may draw out $1,461 to close out the account.

From the output, it is apparent that keeping the car two years is better than buying a car every year, by 12.4 per cent of the differential investment.

The next evaluation compares keeping the car two years as opposed to keeping the car three years (Exhibit XI). One problem exists in this evaluation that did not exist in the first one. The common denominator for the cars must be found, such that a car is sold from each alternative in the same year. Therefore, the Economic Life for this evaluation is six years.

The cost entries for alternative A of this evaluation are the same as those for alternative B of the last evaluation, except for the depreciation offset mentioned during the first evaluation.

Since the economic life of this evaluation is six years, the program does not write off any undepreciated book value until the last year, but rather depreciates the cars over their four-year tax life. How does this affect the input information? The first difference comes in the second year (Exhibit XII).

For the two-year car in the two-year evaluation, $584 was written off and compensated for by placing the same amount in the box which automatically reduces the depreciation claim. In the second year of the six-year evaluation, there is no compensation. Why? Because the program claims this depreciation in the year it would normally occur, and therefore it must be offset in the same year to be economically correct. An examination of the next two years shows that $389 and $195 (Exhibits XIII and XIV) respectively, have been offset for a total of $584, which was written off the books at the time of the sale.

Now consider the cost entries for the three-year car alternative. In the first year they are the same as the other alternative. In the second year, a muffler and tail pipe, a set of tires and wheel alignment (Exhibit XII), add to operating expenses. In the third year (Exhibit XIII), insurance cost is reduced to $185, the rework block includes $25 miscellaneous cost, and rebuild or overhaul includes a $15 battery.

This car will be sold for $840 and it will be written off for the remaining book value of $195. The remaining years are handled similarly.

The output (Exhibit XVII) reflects the total cost difference of $1,601 for six years as opposed to the $1,929 of the first evaluation which only covered two years. The difference in the fixed investments is the price of one car and the difference in the operating expenses is $344 for six years as opposed to $16 for the two-year evaluation.

The cash flow for the first year is exactly zero because the costs for this year are the same. The second year shows an outlay because the two-year plan has revenue from the sale. The next year shows flowback because the three-year plan has some revenue and the two-year plan has purchased the second car. The reasoning is the same for the next three years. The new flowback is $105 for the six years as opposed to the $161 for two.

The first thing noticed in the INV or SAV column is that it is not the same as the cash flow. Because the cash flow alternates between outlay and flowback, the problem of a non-conventional cash flow exists; thus, the possibility of more than one root.

To care for this problem, all deferred outlays are payed off (those which occur after flowbacks) by investing the flowbacks at the cost of capital and using this money to offset the outlays.

Since the outlay in 1965 is not deferred, it is not changed. The flowback of 1966 is completely expended in the attempt to remove the outlay of 1967. The outlay was not removed but since the saving was, it, by definition, is no longer deferred. Some of the flowback of 1968 was used to completely erase the outlay of 1969. The cash flow string is thus converted from non-conventional to conventional.

An important thing to notice is that the net INV or SAV is $146. This is $41 more than the net cash flow and has effectively increased savings. This is economically correct, but care must be exercised not to allow the cost of capital to carry the project (for non-conventional cash flows). In other words, the results must be analyzed in the light of the effect of the cost of capital. The cost of capital could change a small net outlay total in the cash flow column to a small net saving total in the INV or SAV column. This changes the rate of saving from something less than zero to a positive value. Notice, with a cost of capital of 0.0, the net is the same while rate of savings has gone down; that is, the 1.5 per cent cost of capital boosted the rate of savings (Exhibit XVIII).

Again, the rate of saving is calculated by discounting the INV or SAV values until the investments are equal to the savings. The rate of savings of 4.3 per cent, means that the three-year plan is better than the two-year plan by this amount.

The common denominator for the three-year versus four-year plan is 12 years. The four-year car will get new brakes and $75 worth of miscellaneous repairs in the fourth year and insurance drops to $170. Those are the only significant changes. Depreciation is not reduced in the four-year alternate because the car has no book value at the time of the sale.

Exhibit XXXI shows a net difference in the total project costs of $1,829 for the 12-year period and the operating expenses are nearly equal which indicates that the tax credits and resale values of the cars are making the difference.

Although alternate B costs less than A, alternate A is the better project when considering total net cash flow. The three-year plan is better than the four-year plan when considering tax credits. The discounted values are zero because the program does not discount if the INV or SAV column total shows investment. The program will calculate only positive rates of saving between zero and 99.99 per cent.

What has been accomplished by using the program? First, it is known that keeping the car three years is the most economical approach; second, hand calculation and attendant errors have been decreased; and third, calculation time has been reduced.

The values used in these examples are as realistic as could be determined without a very thorough investigation. It should not be assumed that this evaluation will hold true for all types of cars in all investment situations. Each analysis must be made with costs estimated as close as possible to the real conditions.

4.0 CONCLUDING COMMENTS

Each alternate should accomplish the same ultimate goal. It is not reasonable to compare an alternate which produces 100 parts a month to one which produces 200 parts. The alternatives should be balanced so they accomplish the same result, even if one of the facilities must lie idle part of the time or the manpower must be doubled to keep up with the high speed automatic machine of the other alternative.

Another important point is that the rate of savings is not the total answer to the problem. All the values on the output sheet should be considered in the final analysis.

A rate of savings which is low but positive may not signify that the B alternative is the best. Perhaps the company is able to invest money elsewhere at a higher return, in which case choosing alternative A would not tie up the capital in the earlier years of the program. The return on the outside investment would more than cover the outlays in the later years.

This program makes the financial comparison uniformly, accurately, and rapidly. However, evaluation of the output cannot be made by the computer; this must be accomplished by management. Careful analysis and sound judgments therefrom, are of great importance when evaluating the CIP III program output.

COMPUTERIZED ECONOMIC EVALUATION
GENERAL FLOW

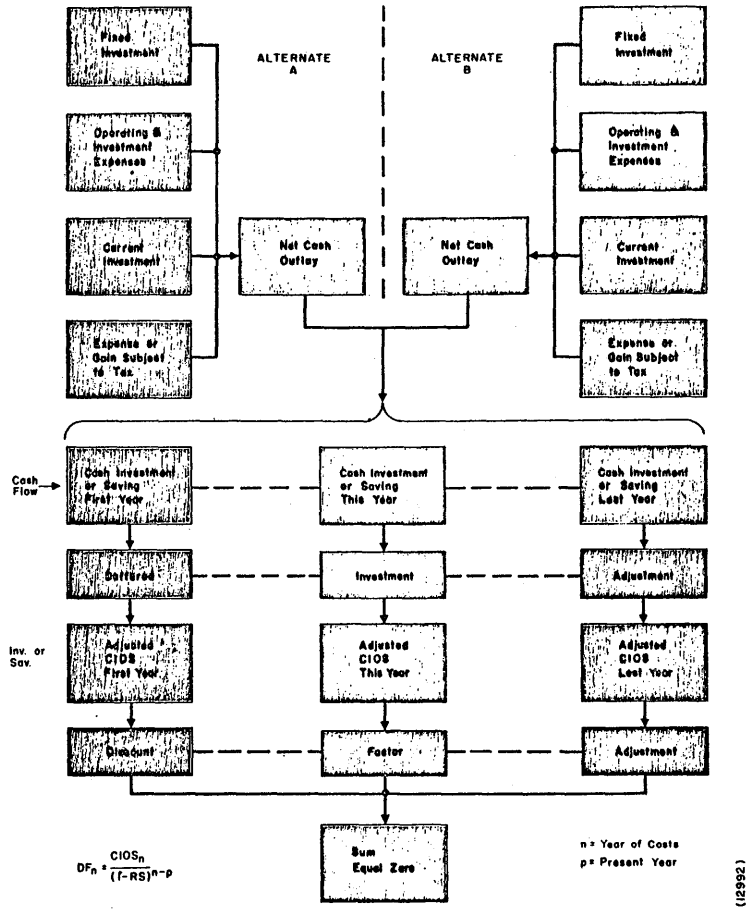


EXHIBIT I

CONTINUOUS STEEL BEAM ANALYSIS AND DESIGN

Stanley W. Woods

STATE HIGHWAY COMMISSION OF WISCONSIN-BRIDGE SECTION

God made a machine, the machine made men, doctors, lawyers, priests, and then, the devil got in and stripped the gears, and turned out the first batch of ENGINEERS.

With this brief introduction to my background I will talk to you about Continuous Beam Analysis as it applies to highway bridge construction. However, several of the ideas expressed are applicable to many types of engineering problems.

The engineering activity on a project goes through four stages: 1. Planning; 2. Analysis; 3. Design; and 4. Construction.

In the Planning stage a structure is studied to determine the economy of different methods of construction. In many cases it may be a determinate structure versus an indeterminate structure. It is probable that an indeterminate structure will have a more pleasing appearance than a determinate structure.

Once the type of structure is determined, it is analyzed to determine the stresses caused by loads acting on the structure. Knowing the stresses, material can be chosen to carry the stresses which is the design. After the design is completed, the structure can then be constructed.

In many cases stages 2, 3, and 4 are a part of the planning stage. It is necessary to analyze, design and study the construction methods before a realistic cost can be established. Also for indeterminate structures, analysis and design must proceed simultaneously because the stresses are dependent on the geometry of the structure.

Let us consider the following stream crossing on an interstate highway as an example. Here is a typical profile of the site:

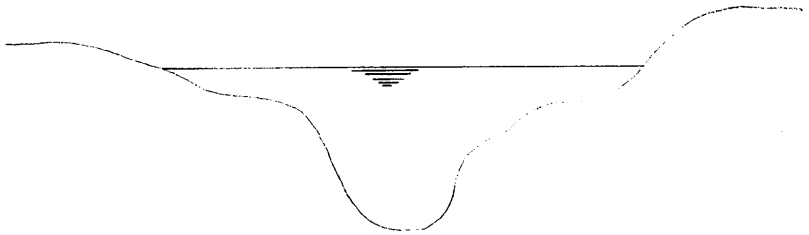


Figure 1

(1)

There are several possibilities on the type of structure that could be used but let us assume that the conditions are such that a steel girder bridge is more economical. Due to the depth of the channel, it appears logical to span the channel. However, what length should the other spans be and should we use continuous or simple spans?

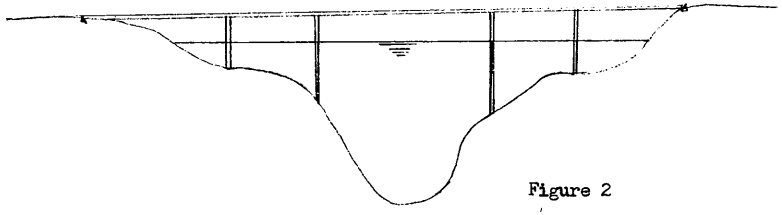
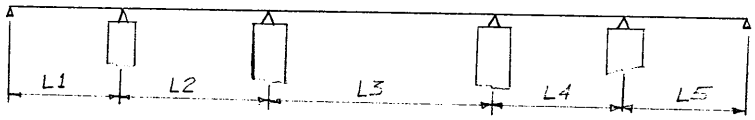


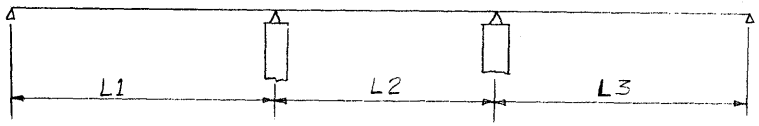
Figure 2

Lets Consider the following possibilities:



CASE 1 (Continuous Spans)

CASE 2 (Simple Spans)



CASE 3 (Continuous Spans)

For purposes of expediency, let us say case 1 is the best solution. Our problem is now to determine the various span lengths and the section required for the girders. For highway girders the girder must carry the dead load of the bridge plus a live load as specified by AASHO.

LANE TRUCK

AASHO HS20 Load

In determining the span lengths, consider the words of D. B. Steinman,

"It is more scientific to eliminate the cause than to build up the structure to resist the effect."

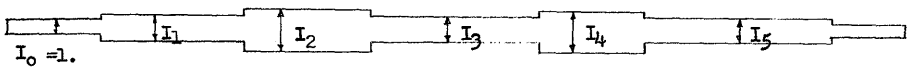
In other words we can choose any span lengths and determine a girder section to resist the stresses. However, by choosing the optimum span ratio, we can balance the stresses to produce a more economical girder section.

We should try cases of various span lengths and analyze each case for the best solution.

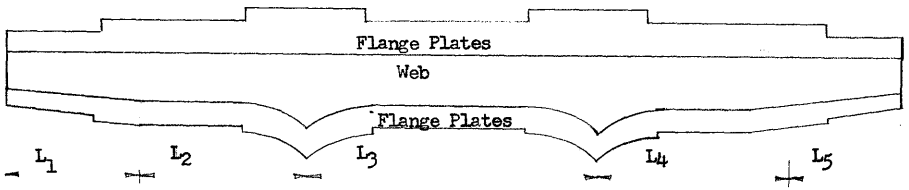
This is where the computer makes us engineers and not technicians. After we make our initial assumptions regarding span lengths and girder sections, the Continuous Beam Analysis program will analyze and design our girder section. This eliminates us from going through a tedious analysis process and then requiring somebody else to check our arithmetic calculations. It should be noted that the computer may not reduce the design time for a specific job but it makes many more trials possible. Since construction costs are the major costs of any job, the additional engineering dollars spent can save thousands of construction dollars by producing the most economical design.

We are now ready to analyze and design our girder. We will make use of the Continuous Beam Analysis and Steel Beam Design program.

For the analysis and design we may assume our section one of two ways:



Variation in Moment of Inertia



Size of Beam Plates

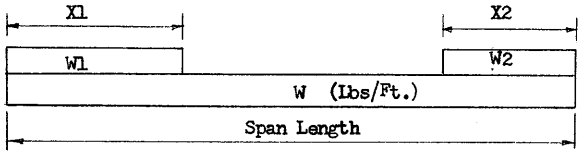
This information is then entered as input data. If the web and plate sizes are given for the section, the moment of inertia variation is computed for the girder.

Using this moment of inertia variation, the concentrated angle change method[1] of describing the elastic slope of a beam is used to find the fixed end moment coefficients, carryover factors, and relative stiffnesses. This method was developed by N. M. Newmark of the University of Illinois.

The computer program will also modify the beam characteristics for intermediate expansion hinges using a method outlined by PCA[2].

These beam characteristics are then substituted into the basic equations of the Slope Deflection method of analyzing structures. This set of simultaneous equations is arranged in matrix form for determining final moments.

The dead load moments and shears are found first. The dead load per span is input as shown:



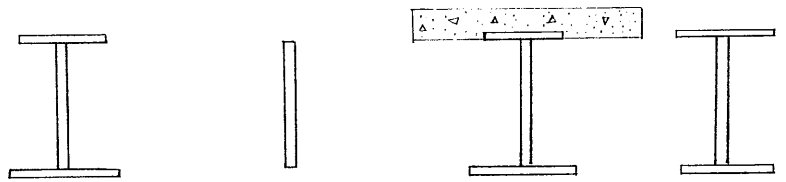
The fixed end moments are found for each span and substituted into the matrix. The simultaneous equations are solved for the final end moments. The dead load moments and shears are then found at the tenth points of each span.

Next, live load moments and shears are found for the girder. Five types of AASHO live load are stored in memory: H15, HS15, H20, HS20, and HS20 Modified. As part of the input data, the type of live load is indicated and the amount of this load that is to be carried by one girder.

The live load specified is then moved across the bridge and the maximum positive and negative moments are found at the tenth points in each span.

The computer now has all the forces acting on the girder and a section can be designed to resist these forces. The input data for the design portion is the allowable stress in the steel, the effective area of slab for a composite section and the allowable concrete stress, the minimum area of the top cover plate, the modular ratio of steel to concrete, the AASHO alternating stress group to be used, and the force the shear connectors will resist.

The girder is designed for composite and non-composite sections. The input section is used as follows:



For purposes of computation the flange plates are assumed to be 16 inches wide. If the negative moment exceeds the positive moment no composite design is made.

Design of the girder sections has become complex in many cases due to modern technology. Repeated loads on a bridge cause fatigue stresses on the girder which may cause failure even though the basic allowable stresses are adequate.

The repeated stresses were always present but were not a factor until high strength steels were developed. Even though the yield point stress of steel was increased, the fatigue stresses causing failure remained the same. To compensate for this the allowable stress of the steel must be reduced to the fatigue stress if this is a problem.

The design portion of this program designs the section for the basic allowable stresses, the alternating allowable stresses, and the welding allowable stresses where splices occur.

This completes the analysis and design done by the computer. The engineer can now analyze the results.

The design flange plate sizes are compared to the input section plate sizes. Since the moments computed are based on the relative moment of inertia of the input section, the design section should be reasonably similar

to the input section. It may be necessary to change the input section and reanalyze the girder if there is too much variation from the design section.

Using this program with our example problem, it is quite easy to try several geometric arrangements to get an optimum design.

This particular program was written for a 20K 1620 computer. It will handle from two to five spans. Five spans is a maximum because of the core memory. The methods used in this program are iteration methods and apply to only one span in most cases. These spans are then connected by using simultaneous equations. It is therefore not too difficult a problem to increase the number of spans if core memory is available. It is also conceivable that with a larger memory the design plate sizes could be used as input data without any interruption from the engineer.

This program has saved our engineers a considerable amount of time. We have used it on as many as 14 continuous spans through proper use of the program.

1. N. M. Newmark, "Numerical Procedure for Computing Deflection, Moments, and Buckling Loads," Paper No. 2202, Vol. 108, 1943 ASCE Transactions.
2. "Beam Factors and Moment Coefficients for Members with Intermediate Expansion Hinges" PCA Bulletin ST 75, 1948.

AUTOMATED STEEL BUILDING DESIGN SYSTEM

L. L. Bathurst, The Rust Engineering Co.

Introduction

This system is a result of the combined efforts of The Rust Engineering Company and I.B.M. It is designed for a 1620 with 60K, two 1311 disks, an on-line printer and a 1627 plotter.

The foundation of the system is the member records which is stored on the second disk pack. All programs take their data from this file and add to or modify it. It is indexed by several keys to allow greater flexibility of use.

All of the programs are stored on the first disk and are under the control of monitor. The system may be added to at any time so long as the basic file record length does not have to be changed. This allows a continual growth of the system by adding programs and sub-programs. We intend to expand its usefullness beyond its present state.

Program "A" Building Descriptiom

To perform analysis and member selection, each member of the structure must be described separately in a "member record". Generation of these records is the function of program "A".

Basic geometry of the proposed structure is set by giving the width of the bays from west to east, south to north and the story heights from bottom to top. Lettering column lines A to Z in the N-S direction, every intersection of a column with a floor becomes a grid point such as "2A3" (second floor, column A3). Every main frame member can be named by two grid points. The program will automatically generate a member record for every member required to fill up the grid described in the input.

In the event a line or plane of framing is skewed, exceptions can be entered. Similar exceptions can be entered to delete a member from the assumed grid or to add a member. Dimensional adjustments are automatically made to all joints in a skewed line or plane.

In addition to a primary framework whose members can be assumed, a building can be expected to contain secondary floor and wall members. Data describing these is organized

within bay boundaries. Typical bays may be repeated without duplicating input data. During processing of secondary framing data, up to 12 special requirements can be set for any member. If none are set, standard values are assumed.

At completion, the program has generated a list of all grid intersections (joints), their locations in space, and the addresses of the member lists associated. The member lists are organized by primary or secondary; and within secondary by horizontal, vertical (N-S), and vertical (E-W). The member lists contain the addresses of the member records.

Program "B" Load Allocation

Two kinds of loads are considered in this program. First, basic kinds of loads such as wind, dead and live loads which affect the design of the primary structure; second, combinations of these loads which are critical for the design of the secondary framing.

In order to compute the strength required for any member, it is necessary to calculate the load imposed on the member by each kind of basic load. It is then necessary to make any combinations which are appropriate. This is the most tedious portion of design in normal circumstances.

In operation, each load in a group is mapped onto a grid of one foot squares. Values within a square are accumulated to produce a map of the total bay loading. Members existing within a bay to be loaded are also mapped onto a grid of one foot squares. Allocation of loads is made by parallel scanning of the two maps to find the members adjacent to a load for each load on the load grid. After complete scanning of the load grid, a second pass is made over the member map to transfer end reactions of beams to supports on other beams, out to the level of the primary framing.

The loads computed for each member are either recorded as "load records" for primary members, or used to compute design values for secondary members. Design loadings on primary members are determined during portal analysis when the combinations of loads are specified.

Member records from program "A" are modified to show the addresses and length of load records for primary members, and to show the maximum shear, moment, and minimum I values for secondary members. Secondary members specified as having continuity at the ends will be computed on the basis of plastic design rules.

Program "C" Portal Analysis

For member selection to be made on a rational basis, the loads applied to the framework by individual members must be analyzed for their interactions. Columns, for example, each increase the load on the column below. Both beams and columns are

affected by the shears due to lateral loading. Prior to initial member selection, assumptions must be made to permit analysis. Portal analysis assumes the floors to be infinitely stiff. It further assumes bay widths to be equal and the structure to be rectangular in its overall outline.

Portal analysis is normally used for tier buildings without diagonal bracing. It will be used, however, for fully braced structures without loss of accuracy.

Input by the designer to this program will specify the bent or bents to be analyzed for a particular combination of loads, and the details of the load combination. Several analyses may be performed in succession.

In operation, the program will compute the maximum values of axial force, shear, moment, and axial force combined with moment for each main frame member, identified by the load combination number and the place on the beam at which the maximum occurs. This information is recorded in the member record.

Program "D" Shape Selection

This program is specific in that it assumes the structure is steel.

The operation of the program is essentially to consult the member records one at a time, and use the parameters set up by portal analysis or load allocation to make a preliminary shape selection. This member is then checked against the AISC design rules in detail for its particular use in the structure. The process is repeated if necessary until an acceptable shape is found in the catalog(stored on the system disk). The catalog may be divided into as many as 9 selection groups. The designer has control of the selection. He can limit selection to any set of groups or even pre-select a member.

Upon completion of member selection the designer will obtain a complete tabulation of the members selected, by strength required, and shape furnished.

Program "E" Output Options

After execution of any of the preceding programs, the input stack may contain a control card set to produce one or more reports from the data contained on the working disk pack.

It may also be desirable to edit the contents of the disk pack to reflect arbitrary decisions, or to copy typical data from member record to member record.

The plotter is used to make Engineering drawings of the building as input and finally as designed. This gives the designer a visual check on his input and the design.

THE OHIO STATE 1620 PROGRAMMING SYSTEM

By

James P. Shaffer, Computer Center
The Ohio State University
Columbus, Ohio

ABSTRACT

The Ohio State University Computer Center has implemented a unified loader-assembler-compiler programming system on a 20K, card oriented IBM 1620 computer. The system features a single relocating loader, standard calling sequences and a general purpose input/output package. The main program may be written in either compiler or assembler language.

Notable features of the loader include:

a) the availability of eight loading increments plus absolute loading;

b) the loading of any aggregation of assembler and compiler object decks;

c) a typed memory map of the loaded program; and

d) the use of library routines written in either assembler or compiler language.

The system assembler, OSAP II, accepts a language which is basically SPS and features:

a) relocatable or absolute assemblies;

b) arbitrary (parenthesis free) address expressions;

c) an iterative partial assembly procedure; and

d) preservation of the input format.

The system compiler, 1620 SCATRAN, (whose source language is somewhat like FORTRAN IV or MAD) features:

a) arbitrary subscript expressions;

b) externally compiled subroutines;

c) a more powerful iteration statement; and

d) a conditional statement.

The language is completely upward-compatible to SCATRAN for the IBM 7094 while the compiler itself is upward-compatible to a larger 1620.

ADMINISTRATION OF A SHORT COURSE IN FORTRAN

by

James R. Wright

A Paper

For presentation at the Joint Western and Mid-Western Regional Meeting of the 1620 Users Group

Norman, Oklahoma

Nov. 9, 1964

I. INTRODUCTION

This paper contains some quizzes and a final exam which were used in teaching Fortran at the Trane Company, La Crosse, Wisconsin. Neither the tests nor the classes were the ultimate refinement in teaching techniques; but they are a beginning, and if they can be of use to others, the purpose of this paper will have been achieved.

II. REASONS FOR GIVING THE COURSE

The objective was three-fold:

1. To awaken engineers and others to the possibility of machine computation of their specific problems.
2. To publicize the limitations of the computer and define the types of jobs for which the computer is most suitable.
3. To enable personnel to read and understand programs written for them.

The engineering computer at Trane is run only by computer section personnel. However, once a program is written and run for an engineer, it has been found very advantageous for that engineer to be able to read and understand his program, especially in cases of explaining the operation of the program to code committees, or to new men in his group, or even in recommending changes to be made in the program.

Therefore, the lesson content was directed more toward teaching personnel to follow existing programs rather than create new ones.

III. PROCEDURES

Notification of the class was sent to all vice-presidents who circulated the notice and advised the computer section of desired

enrollments. Enrollment was limited to 30. Each course included 50% to 75% engineers. Each course ran for 5 weeks, one afternoon a week from 4:00 PM to 6:00 PM. The end of the normal working day is 4:30 PM.

The first two classes were given without the Programmer Aptitude Test. The bewildering performance of 2 or 3 individuals in each class prompted the administration of the P.A.T. at the outset of all subsequent courses. Thirty copies of the Revised Programmer Aptitude Test by J. L. Hughes and W. J. McNamara were borrowed from IBM, and given to the applicants for the third class as an entrance requirement. The fourth class took the P.A.T. as a portion of the first session.

Tables were set up for the administration of the P.A.T.; all other sessions used only auditorium chairs. A chalk board and lecturn were also used. The book used was Form C26-5619-3 (1620 Fortran with Format). Books were supplied by IBM.

IV. THE QUIZZES

The purpose of the quizzes is two-fold; to test the student's knowledge of the subject, and further the teaching process by pointing out his areas of weakness. All quizzes were graded, correct answers supplied for each question missed, and handed back to the students.

Quiz No. 1 was directed toward solidifying definitions of fixed and floating point variables and constants. It was graded on the basis of 30, with one point for each question. Ten to 15 minutes was allowed.

Quiz No. 2 tested the awareness of priority of operations, mode changing across an equals sign, etc. Top score was 100 with 10 points for each correct answer. Presence or absence of a decimal point was graded on all problems except the last, on which the infinity sign was also accepted as 100% correct. Time was 15 to 20 minutes.

Quiz No. 3 was the only one that concentrated on the student's ability to create correct Fortran statements. There are 12 errors in the 10 statements. Five points were given for locating each error, five for stating what the error was. The students were told that there were 12 errors and given 15 to 20 minutes for the quiz.

Quiz No. 4 was given as a take-home exercise. Since there were so few answers required, the grades were not considered indicative of the student's work and were not recorded and correlated with the other tests. This quiz was discussed thoroughly in class.

V. THE FINAL EXAM

The copies of the quiz handed back were kept by the students. The final exam was not. The main reason for this was that the final was designed to be taken on P.A.T. answer sheets, and graded with the P.A.T. grading key. It was desired to keep the access to this key very limited and under strict control. Those using this exam will want to observe the same policy.

Thirty-five copies of the test booklet were made and numbered. The time allowed was 30 minutes after which all students were asked to put the number of their exam booklet on the answer sheet. Then they were instructed to erase all existing pencil marks in the test booklet. The booklet and answer sheets were then collected and

graded. Total score was rights minus 1/4 of the wrongs as in the P.A.T.

The final was written for Fortran with Format and used the fact that upon a normal exit from a DO-LOOP the index is usable and is higher than the maximum value specified in the DO statement.

VI. CORRELATIONS and CONCLUSIONS

Correlations were run only on the students who took the P.A.T., the first three quizzes and the final exam. This eliminated all but 24 in the 3rd class and 10 in the 4th since many students missed at least one quiz, especially in the 4th class which was taught during the vacation season in August. The correlation coefficient between the P.A.T. and the final exam was .707 for the 3rd class and .354 for the 4th class.

All of those completing the class were able to read, understand and explain programs written for their use. Furthermore, 21 of the 82 people having completed the course to date have, on their own, written and debugged one or more useful programs which would otherwise have been written by Engineering Computer Section personnel at such time in the future as they could be scheduled.

The objectives of the course have, therefore, been achieved.

AN ANALOG-TO-DIGITAL CONVERSION SYSTEM FOR THE IBM 1620, MODEL I

W. WILCOXSON
and
A. JACKSON
Port Hueneme, California

INTRODUCTION

This paper describes the Analog-to-Digital Conversion (ADC) System for the IBM 1620, Model I, at the U. S. Naval Civil Engineering Laboratory. The ADC is a part of the data handling complex for the atomic blast simulator at the Laboratory. It was designed for rapid, accurate, efficient, economical and timely reduction of data generated by that facility.

Although the purpose of this paper is primarily concerned with the ADC, a superficial treatment of the data handling complex is presented. This is given to illustrate a non real time application of an analog-to-digital converter and to show the motivation for fabricating the ADC in the first place.

A FORTRAN compiler was produced to provide the non-professional programmer with simple and efficient software for using the ADC. Although a cursory treatment is given of the hardware, additional specifications and modifications to FORTRAN I are in sufficient detail for implementation of the ADC language.

A data reduction program for the data handling complex is included to demonstrate the use of the ADC FORTRAN and to provide data for actual comparisons between the automated and manual methods.

DATA HANDLING COMPLEX

Data Collection

Various types of transducers may be used to provide a voltage input to the data collection system. Output voltage of the transducer may be as low as 2 millivolts for full scale deflection. Although the system is designed for operation with a wheatstone bridge, any self-generating and potentiometric sensor may be used.

The magnetic tape recording system is an FM (frequency modulated) data collection device. Seven magnetic recording heads (channels) are provided. Each head is driven by its own direct-record amplifier. Speed of the transport for the 1/2-inch tape is maintained at 60 inches per second.

Figure 1 is a schematic of the data collection system. Input to the FM data system is from a DC transducer. The signal from a transducer is balanced, step attenuated and then amplified. An oscillator (VCO) is controlled by the amplified voltage. The frequency of the VCO is proportional to the controlling voltage. A signal to the VCO may be up to 1000 cycles per second. Outputs of as many as 18 VCO's can be summed and recorded as a composite signal on one channel of the magnetic tape.

Data Conversion

Information stored on the magnetic tape must be converted for output on an oscillograph, pen recorder, oscilloscope or to a digital computer. Figure 2 is a schematic of the data conversion and reduction system. Test data stored on a particular channel of the magnetic tape is selected and played back. The speed of the magnetic tape reader determines whether the time base is real, compressed or expanded. Selection of the playback amplifier and band-pass filter determines which frequency of the composite signal will be demodulated by the discriminator. Output current of the discriminator is proportional to the original transducer stimulus.

The data conversion system, including the analog-to-digital sub-system, is an input device to the IBM 1620 Data Processing System. Figure 3 is a component block diagram of the ADC. Control of the ADC is by both the computer and a "key" signal from the playback amplifier. Format for the input signal is given in Figure 4. The range of the input voltage is between -9.99 volts and +9.99 volts. This voltage is digitized with a maximum absolute error of 0.01 volt. Although the ADC was designed for the data conversion system, any signal satisfying its input requirements may be used. The ADC utilizes the paper tape reader channel (1621 interface) and is selected by a Read Numeric from Paper Tape (RNPT) instruction.

The ADC subsystem contains programmed hardware that executes a fixed sequence of instructions. A "key" of exactly 8-cycles of a 1-KC signal initiates the ADC after the ADC is selected by the computer. A sample of the calibration voltage is converted to a 3-digit number with field flag and sign, and si then transferred to the computer. After the time period set for the start delay, variable from 0.1 to 1000 milliseconds, samples of the data voltage are digitized at a rate of approximately 4000 per second until the transferred data sample count is satisfied. The time interval between transferred samples is determined by the transfer rate. Every sample, every other one, every fourth one, every eighth one or every sixteenth one is transferred according to the setting of the transfer switch. This gives approximate sampling rates of 4000, 2000, 1000, 500 and 250 samples per second. After the given number of samples are transferred to the computer, a stop delay is initiated for 0, 0.1, 0.2, 0.4 or 0.8 seconds. Immediately following the stop delay, a single sample of the "steady state" voltage is digitized and transferred. The ADC generates a disconnect for the 1620 (end-of-line or record mark) which releases the computer from the ADC and allows the computer to proceed to the next program step. After the disconnect, the ADC subsystem resets and waits for another call from the computer.

Data Reduction

Several extensions to FORTRAN were made to give the scientist, engineer or non-professional programmer a simple and effective language for communicating with the ADC system. The additional specifications and the required changes are included in Appendix A.

An elementary data reduction program for the blast simulator data handling complex is attached as Appendix B to illustrate an application of the ADC and the ADC FORTRAN. The program will process data generated by acceleration, velocity, displacement, strain and pressure transducers. For 500 data samples of each gage reading, the time required for the data reduction program to summarize the data for each gage is about 45 seconds. Velocity, displacement, moment and thrust tabulations are at 80 points per minute.

DISCUSSION

Actual comparisons between the automated and the manual methods are difficult to make. The time required to reduce a set of data by the manual method varies with the individual and the quality of the record produced by the oscillograph. If the traces on the oscillograph are badly mixed, then the time to reduce the data is greatly increased.

One case history is available for comparing the automated and manual methods. A technician who is recognized for his efficiency and good work habits required three weeks to reduce a set of 36 gage readings. Now the same data reduction can be accomplished by the data handling complex in an hour. An average individual reducing the data would be expected to require four weeks. In general, the time required should range from three to five weeks for 36 gages.

Many of the potential sources for human error in the manual method do not exist in the automatic data reduction process. In the manual mode, each measurement of the amplitude and time of a trace is subject to error. The possibility of an error is increased if the traces on the oscillograph are mixed. Each arithmetic operation by the manual method is a source for error. No parallels to these sources for error exist in the data handling complex.

The data handling complex represents several advances in technology. No known system existed for rapidly sampling a single analog input and storing the sample in the memory of the IBM 1620. The rate of 12,500 digits per second is 24 times greater than the input rate from paper tape and 36 times greater than that from cards for the Model I.

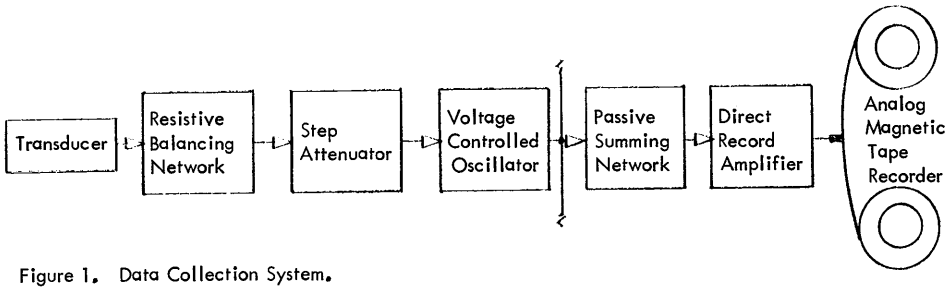


Figure 1. Data Collection System.

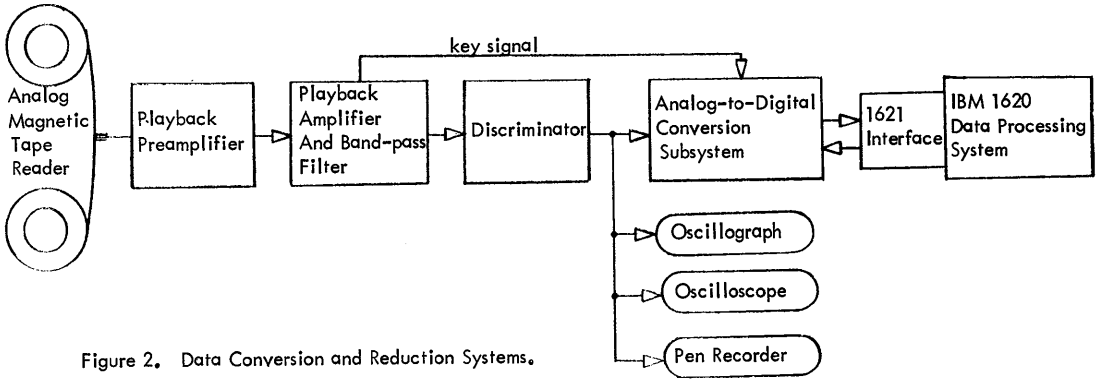


Figure 2. Data Conversion and Reduction Systems.

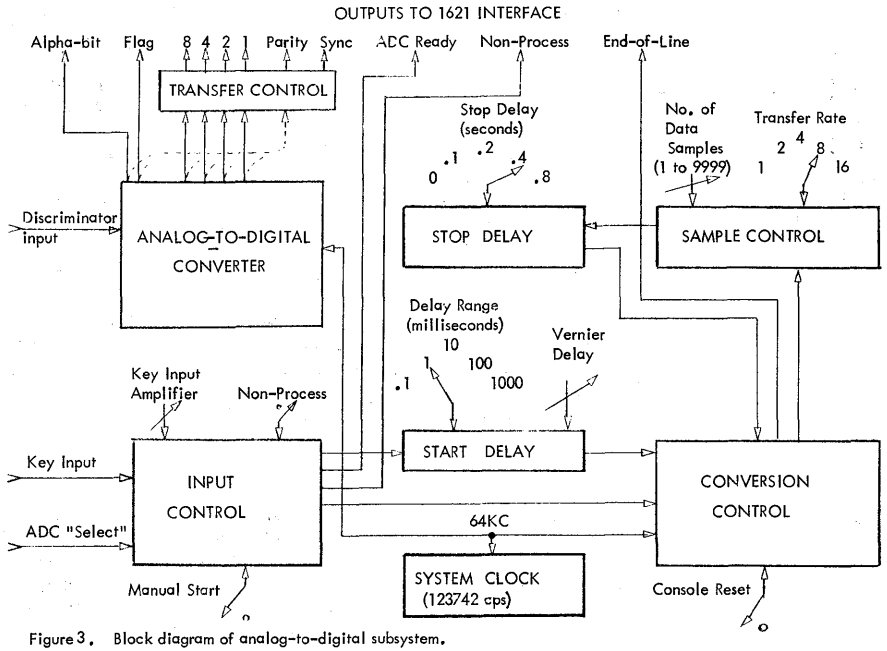


Figure 3. Block diagram of analog-to-digital subsystem.

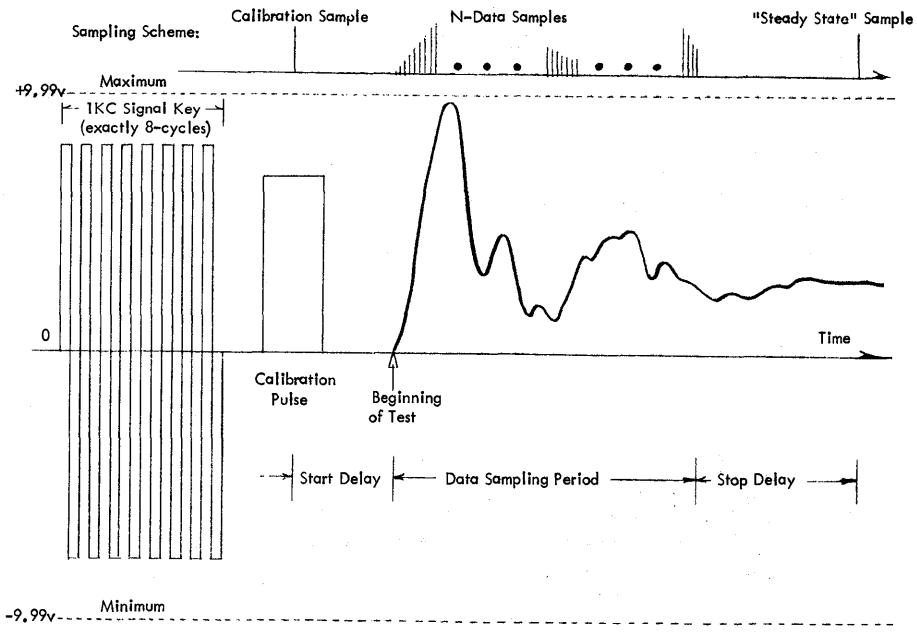


Figure 4. Format of input signal to the analog-to-digital subsystem.

A MULTIVARIATE ANALYSIS PACKAGE

By
Rex L. Hurst
Computer Center
Utah State University

Introduction

There is a growing trend in Computer Science toward programming systems rather than single programs. A number of authors of statistical programs are attempting to combine these programs into packages that will provide a wide variety of statistical services. Mr. Bohles (2) has done an outstanding job in providing a series of programs for handling multiple linear regression systems. Drs. Cooley and Lohnes (3) have provided an unusually competent set of programs for handling multivariate analyses on large scale machines. Dr. Lohnes has also developed a series of programs for the IBM 1620 which are available through private communication. This paper is directed toward further emphasizing the importance of developing programming systems rather than individual programs.

Many techniques in the field of multivariate analysis use essentialy the same basic information matrices. Further, experience has shown that more than one technique is usually applied to the same set of data. It is desirable, therefore, to split the data collection phase from the analysis phase in developing a multivariate package.

The aim is to develop a multivariate data collection program that is versatile enough to provide the basic information matrices for a wide variety of multivariate techniques. It is hypothesized that the following multivariate techniques can all be based on the same multivariate data collection program:

* Multiple Regression
Partial Correlation
* Factor Analysis
* Canonical Correlation
Discriminant Function Analysis
Covariance Analysis -
Completely Randomized Design
Multivariate Analysis of Variance -
Completely Randomized Design

The techniques provided for in this package are starred. A number of computer programs by others and by myself, either have been or might be modified to fit in this package (5,8,9 10,11,15).

Multivariate Data Collection

The multivariate data collection program (MDC) will perform both WITHIN and AMONG group calculations. Table 1 gives a listing of the various types of output that it will produce. Card types 6,7,8 are produced only if multiple data groups are used.

Table 1.

Output of Multivariate Data Collection

Column One	Description of Card Types
b	Control Card
1	Variable Specification Cards
2	Group Size Cards
3	Within group means and standard deviations
4*	Within group corrected sums of squares and products
5*	Within group correlations
6	Over-all means, pooled (error line) standard deviations
7*	Pooled (error line) sums of squares and products
8*	Pooled (error line) correlations

* Punched out under internal switch control

MDC uses the forcom subroutines (14,16) to provide flexibility of card forms and versatility in the formation of variables to be used. The forcom subroutines have been added to Fortran to facilitate card handling. Six of the fourteen subroutines in the package are especially useful. The forcom subroutines allow the fortran programmer to read a card into the computer and store it as a card image; the programmer may then have repeated access to the card or cards, and if desired, he can organize new card forms.

The six most useful forcom subroutines are the CDS, RCD, GET, PAS, PUT , and PCH. The CDS specifies the number of card types that are going to be worked within the program. The RCD simply reads a card and stores it in one of the card images. The GET allows one to go to a card image, extract a variable and use it as a Fortran variable. The PAS allows information to be moved from one card image to another to affect reorganization of card images.

The PUT stores a Fortran variable in a card image. The PCH is used to output the card image as a card. The best discussion of these forcom subroutines is contained in Reference 16. Reference 14 was written to be a supplement to the original forcom subroutine package written for Fortran 1. Fortran 1 has subsequently been dropped from the library.

In the development of MDC, three card images were assumed to be adequate for most jobs. Algorithm 1 provides the flexibility necessary to read in a research record contained in one, two, or three card forms. The variables may be extracted at random from each of the three possible card forms.

Algorithm 1.

Card Reading, Transformations, and Calculations

```
      DØ  6 K = 1, NØBS
      DØ  7 M = 1, NCRD
      ZA = M
    7 ZB = RCD (ZA)
      DØ  6 I = 1, NV
      ID = IX(I)
      GØ TØ (21,22), ID
   21 X(I) = GET(F(I,1)) = CØN(I)
      GØ TØ 30
   22 X(I) = GET(F(I,1))*GET(F(I,2))-CØN(I)
   30 SUM(I) = SUM(I) + X(I)
      DØ 6  J = 1,I
    6 A(J,I) = A(J,I)+X(I)*X(J)
```

The Forcom Arguments necessary for forming the various transformed variables desired by the user, have been read in before this algorithm occurs in the program. The nine possible transformations so far provided are indicated in Table 2.

Table 2.

Transformations Available in MDC

Code	Transformation
1	$A_i \longrightarrow X_i$
2	$A_i B_i \longrightarrow X_i$
3	$A_i B_i C_i \longrightarrow X_i$
4	$A_i B_i C_i D_i \longrightarrow X_i$
5	$A_i^{1/2} \longrightarrow X_i$
6	$\text{Log}_e (A_i) \longrightarrow X_i$
7	$1/A_i \longrightarrow X_i$
8	$A_i - B_i \longrightarrow X_i$
9	$A_i/B_i \longrightarrow X_i$

The desired transformations are spelled out by means of a variable specification card. Besides the desired type of transformation, the card provides for up to four Forcom Arguments which are used to effect the desired transformation. For example; a cubic term would require three Forcom Arguments, one for each variable used in the cubic term. A difference would require two arguments, a logarithm, a single argument, etc. The transformation is actually made in the program by means of a computed - GO TO - based on the transformation code. When the code is referenced, the computed - GO TO - branches the program to the particular Fortran function which will produce the transformed variable using the series of Forcom Arguments provided in the Forcom Argument Matrix. Only the linear and quadratic transformations are illustrated in the algorithm. Any other transformation that is desired by the user may be added by writing the Fortran statement for the transformation in terms of four possible Forcom Arguments.

The computation of sums of squares and products of variables, using Fortran, is beset with many problems. The product of two 4-digit numbers is either a 7 - or eight-digit result. A sum of 4-digit products, will produce over an 8-digit result. In Fortran, this means that the least significant digits of such a sum have been truncated. To make matters worse, the corrected sum of squares or products is usually produced by subtracting a correction term

from the uncorrected sum of squares or products. Those who have done this operation on a desk calculator should have been impressed by the fact, that the correction term is generally of the same order of magnitude as the uncorrected sum of squares. Often, the first two digits of these quantities are identical. When using Fortran to obtain a difference between two such 8-digit numbers in which the first one or two digits of each number are identical, you are left with a 6-digit difference with the two trailing positions being filled in with zeros. The only satisfactory answer to this problem, of course, is to go to extra precision arithmetic.

One partial solution, however, has been provided in the program. Many of you would know this technique as using an assumed mean. If a set of data is perused, a rough approximation of the mean may be guessed when approximation of the mean is subtracted from each observation. The numerical magnitude of the variable is often reduced. The following equations show that if a constant C is subtracted from a variable Y to obtain a new variable P, certain interesting properties result:

$$
\begin{aligned}
P_i &= Y_i - C \\
\Sigma P &= \Sigma Y = nC \\
\bar{P} &= \bar{Y} - C \\
\bar{Y} &= P + C \qquad (1) \\
\Sigma(P-\bar{P})^2 &= \Sigma[Y-C) - (\bar{Y}-C]^2 \\
\Sigma(P-\bar{P})^2 &= \Sigma(Y-\bar{Y})^2 \\
\Sigma P^2 - (\Sigma P)^2/n &= \Sigma Y^2 - (\Sigma Y)^2/n \qquad (2)
\end{aligned}
$$

The average of Y can be obtained by adding the constant to the average of the new variable P, equation 1. The corrected sum of squares for the variable P is identical with the corrected sum of squares for the variable Y. From equation (2), if $(\Sigma P)^2/n$ is sufficiently small, the floating point subtraction $\Sigma P^2 - (\Sigma P)^2/n$ can be effected without any loss of computational accuracy. The object, then, of this arbitrary constant, is to find a constant that is sufficiently close to the arithmetic mean of the observations so that when subtracted, it will reduce the numerical magnitude of the variable enough so that the term $(\Sigma P)^2/n$ is of an order of magnitude smaller than the summation of ΣP^2. For example; if the data consisted of the numbers, 1013, 1015, 1047, 1055, it would be expedient to use 1030 as an assumed mean. This would give

Algorithm 3.

Space Utilization

```
      DIMENSIØN A(40,39)
      DØ  8  I = 1, NV
      X(I) = SUM(I)/ØBS
      AVE = X(I) + CØN(I)
      DØ  9 J = 1, I
      A(J,I) = A(J,I) - SUM(I)* X(J)
    9 A(I+1,J) = A(I+1,J) +A(J,I)
    8 PUNCH 104, JØB,I,SUM (I), AVE
```

To reference the lower half of the matrix, the element is identified as $A_{i+1,j}$ rather than $A_{j,i}$. The sequence in which the algorithm has been written dictates that the upper triangular portion of the matrix is developed by columns rather than rows. This allows averages and averages augumented by a constant to be computed in the same algorithm. If the upper portion is referenced by rows, the bottom portion will be referenced by columns.

Programs Based on MDC

In developing the programs which utilize the multivariate data collection output, two guide lines have been set up. Programs should, as much as possible, use the output of the multivariate data collection program or previous program without necessitating extensive operator decisions. Great versatility should be provided in selecting the variables to be used in a particular model.

In many of the programs which follow MDC, the first step after reading the respective control card is to read a vector of elements. This vector defines the desired variables and their order. This vector is stored internally as the vector ID_k. Algorithms 4 and 5 have been developed to allow the user to extract the particular combination of variables required for the various models that he may wish to use. Algorithm 4 is a single dimension selection algorithm. Algorithm 5 is the two-dimensional selection algorithm.

the desired result with respect to the floating point subtraction and would produce fewer digits to be manipulated in the computation.

For those who object to the use of Forcom, Algorithm 2 provides the same computations as Algorithm 1. It does not provide for selecting variables at random from cards, nor does it provide for transformations. The A type conversion may be used to obtain flexibility of Format. The subtraction of the constant is achieved as in the Forcom Algorithm.

Algorithm 2.

Forcom Replacement

```
    DØ  6 K = 1, NØBS
    READ 120, (X(I), I = 1, NV)
    DØ  6 I = 1, NV
    X(I) = X(I) - CØN(I)
    SUM (I) = SUM(I) + X(I)
    DØ  6 J = I, NV
6   A(I,J) = A(I,J) + X(I)*X(J)
```

One of the most severe restrictions that the programmer faces in trying to use the 1620 to develop programming systems, is in space availability. Those familiar with sums of squares and products calculations realize that only the upper triangular portion of this matrix is required because of symmetry. Efficient utilization of space can be realized by storing this upper triangular matrix in terms of a single dimensional array. This, however, requires extensive address calculations inside the basic data calculation loop, and in general will slow the program down excessively. Two types of basic calculations are required in the multivariate data collection program: 1) the WITHIN group calculations, and 2) the AMONG group calculations. If the one dimensional approach had been used, this would have required the definition of two matrices with the problems involved in address location. It was decided, however, that the problem could be solved by defining a (K+1) by K matrix, using the upper triangular portion for storage of one matrix, and the lower portion offset by one row to store the other matrix. This is illustrated in Algorithm 3.

Algorithm 4.

Single Dimension Selection Algorithm

```
    DØ  60 I = 1, NV
    READ 501, ELEM
    DØ 61 K = 1, NVAR
    IF(I-ID)K)) 61, 62, 61
61  CØNTINUE
    GØ TO 6Ø
62  SUM (K) = ELEM
60  CØNTINUE
```

Algorithm 5.

Two Dimensional Selection Algorithm

```
    DØ  50 I = 1, NV
    DØ  50 J = I, NV
    READ 500, ELEM
    DØ 51 K = 1, NVAR
    IF (I-ID(K)) 51, 52, 51
51  CØNTINUE
    GØ TØ 50
52  DØ 53 L = 1, NVAR
    IF (J - ID(L)) 53, 54, 53
53  CØNTINUE
    GØ TØ 50
54  IF (K-L) 55, 56,55
56  A(K,L) = ELEM
    X(K) = SQR(A(K,L))
    GØ TØ 50
55  A(L,K) = ELEM
    A(K,L) = ELEM
50  CØNTINUE
```

These selection algorithms allow the user to select any possible subset of variables including the entire set in any possible order. Algorithm 5 assumes that the matrix desired is being read a row at a time as the upper triangular portion of a symmetric matrix.

Multiple Regression

Three programs have been developed to handle multiple regression problems. The first is the basic multiple regression program, the second is a program for computing sums of squares to subsets of coefficients; and the third is a program for computing plot back.

The multiple regression program is written so as to do all of the matrix operations within one defined matrix area. Figures 1 and 2 indicate before and after status of the matrix area upon using the matrix inversion algorithm.

Figure 1.

Matrix allocation before inversion and solution

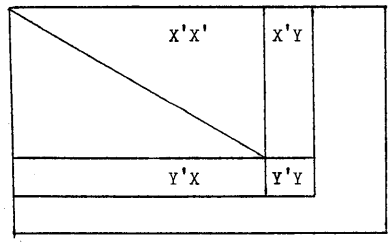
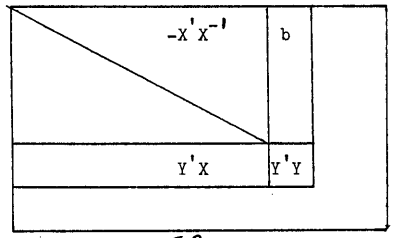


Figure 2.

Matrix allocation after inversion and solution



The program is written to handle any basic subset of the MDC output and to handle multiple dependent variables. The series currently does not include a program that provides stepwise multiple regression.

To use the multiple regression program, it is necessary to prepare a control card spelling out the job identification, the number of independent and dependent terms, and whether or not the inverse is wanted as output. This control card is followed by a series of variable selection cards which use Algorithms 4 and 5 to select the proper elements out of the output of the MDC Program. The multiple regression program uses the MDC output through card type 4.

The output of the multiple regression program consists of a duplicate copy of the control card and the selection vector followed by the inverse when desired. This is followed by a regression analysis for each of the dependent variables. The regression analysis consists of the mean squares for each of the individual coefficients, the partial regression coefficients, and the standardized regression coefficients. It also gives the mean square due to the model, mean square due to error, and the multiple R^2 for fitting the model.

The "Sums of Squares Due to Subsets of Coefficients in Multiple Regression" program is used to test the hypothesis that groups of the partial regression coefficients are simultaneously equal to zero. A discussion of the methodology used in this program and the interpretation of the results is given in (12). The input consists of a control card spelling out the job identification, the total number of coefficients that are to be used in all subsets that the user desires to investigate, and the total number of subsets that the user wants to use. This is followed by a selection vector identifying the coefficients that are going to be involved in any of the subsets. The entire output of the multiple regression program follows this control card and selection vector. Subset size trail cards that define the number of coefficients to be used in each of the respective subsets follow the multiple regression output. Next is a subset selection vector card, for each subset, which identifies each coefficient in the subset. The output consists of an analysis for each of the subsets that gives the degrees of freedom, and the sum of squares and mean square due to the subset. Besides the analysis, all of the control cards and selection vectors etc., are punched out preceding the analysis.

The plot-back program uses the entire output of the multiple regression program followed by the entire input to the multivariate data collection program. The output of the plot-back program will be one card for each dependent variable for each observation. This card will list the value of the dependent variable, the predicted value of the dependent variable, and the deviation (actual value minus predicted value).

Factor Analysis

Two programs in the package are concerned with factor analysis. The first of these is a principal components factor analysis program. The second is the Varimax Matrix Rotation Program, 6.0.094, of Mr. Teeples. This program has been modified to accept the principal components output.

The factor analysis program input consists of a control card spelling out the job identification, the number of variables to be used, an internal switch value to control the statistical tests performed and the punchout options, and the number of observations. This is followed by a variable selection vector that defines variables to be selected from the MDC output. The MDC output used for this program is any of the correlation matrices that it produces, card type 5 or card type 8. The number of factors that are extracted is operator controlled with sense switch options. At the completion of each root, the magnitude of the root is typed out so that the operator may decide whether to continue with root finding.

The output of this program consists of the characteristic roots for each of the factors, the loading factors for each of the individual variables, and the residual correlation matrix. Statistical tests are performed , and the reproduced correlation matrix is punched out under internal switch control. The varimax rotation program accepts the complete output of the factor analysis program as input. A control card is necessary to spell out the job identification, the number of factors that have been extracted, and the tolerance value to be used in internal calculations. The output is a rotated vector for each of the factors produced by the factor analysis program. Good discussions on the applications of factor analysis are given by Harmon (7) and by Cooley and Lohnes (3). Biological applications are discussed by Ferrari and Venekamp (4) and by Goodall (6).

Canonical Correlation

The two canonical correlation programs have been extracted almost entirely from the author's own program (10). It must be noted that the original

programs contain a computational error. Only the first canonical correlation may be correct. These programs will be re-submitted to the library with the necessary corrections.

The canonical correlations program will accept any of the correlation matrices produced by the MDC Program. A control card spells out the job identification, the number of left- and right-hand variables, and the total number of observations. The customary selection vector then defines the variables that are to be selected from the MDC output for the particular mathematical model. The first program in this series reorganizes the correlation matrices to produce the R_{11}, R_{12}, R_{22} matrices, the inverse of R_{11} and R_{22}, and the matrices that have to be used in finding the characteristic roots and vectors. The second program accepts the complete output of the first, and outputs the canonical correlations and the left and right hand weights for each of the individual canonical correlations. The residual matrix is punched out to enable the user to evaluate the sufficiency of the number of correlations that have been extracted. The number of correlations that are extracted is operator-controlled by means of sense switches. Statistics are punched out to enable the user to evaluate the statistical significance of the results. A good discussion of the canonical correlation procedure is contained in Cooley and Lohnes (3).

SUMMARY

This series of programs has been developed to serve as a basic multivariate analysis package to be used in a statistical laboratory. The package is now only partially completed, and other programs will be added in the future. The completed portion will be available the first quarter of 1965. Other computing centers and statistical programming groups should add to and alter this set of programs as desired. The author does hope, however, that anyone making such modifications will keep him informed as to their progress.

REFERENCES

1. Anderson, T.W., "An Introduction To Multivariate Statistical Analysis," John Wiley and Sons, 1958.

2. Boles, J. N., "80-Series Multiple Linear Regression System," 06.0.143.

3. Cooley, W. W., and Lohnes, P.R., "Multivariate Procedures For The Behavioral Sciences," John Wiley & Sons, 1962.

4. Ferrari, T. J., and Venekamp, J.T.N., "Factor Analysis in Agricultural Research," Netherlands Jour. Agr. Sci., 5: 211-21, 1957.

5. Garber, M.J., "Stepwise Regression," 06.0.031.

6. Goodall, D.W., "Objective Methods for the Classification of Vegetation," Australian Jour. of Botany, 1954.

7. Harmon, H.H., "Modern Factor Analysis," The University of Chicago Press, 1960.

8. Hurst, R. L., "Partial Correlations," 06.0.064.

9. Hurst, R. L., and Wiser, G., "Discriminant Function Analysis," 06.0.076.

10. Hurst, R. L., "Canonical Correlation," 06.0.137.

11. Hurst, R. L., "Multivariate Analysis of Variance - Completely Randomized Design," 06.0.138.

12. Hurst, R. L., and Pederson, M.W., "Alfalfa Seed Production as a Function of Genetic and Environmental Characteristics," Advancing Frontiers of Plant Sciences, 8: 41-54.

13. Hurst, R. L., "Statistical Techniques Which Might be Useful in Further Research," Weather and Our Food Supply, CAED Report 10, Iowa State University, 1964.

14. Pope, W. L., "Forcom Subroutines for Fortran W/Format," 01.6.051.

15. Teeples, T. C., "Varimax Matrix Rotation," 06.0.094.

16. Webster, J. S., "Forcom Subroutines For Fortran II," 01.6.074.

17. Williams, W. T., and Dale, M. B., "Partition Correlation Matrices for Heterogeneous Quantitative Data Nature," 196: 602-3, 1962.

"PACTOLUS"
A DIGITAL ANALOG SIMULATOR
FOR THE IBM 1620

by
R. D. Brennan
H. Sano

International Business Machines Corporation
Research Laboratory
San Jose, California

This paper was presented at the 1964 Fall Joint Computer Conference in San Francisco on October 28, 1964. It is published as part of the AFIPS Conference Proceedings, Volume 26, Part 1, 1964 FJCC.

ABSTRACT:

A number of digital computer programs have been developed for simulating the operation of analog computers. Primarily, they were attempts to combine the computational ability of the digital computer with the structural organization of the analog computer. The block-oriented languages used with these programs did permit the representation of analog elements and their interconnection; none of these previous programs, however, had any real provision for the operational flexibility which is the forte of analog simulation. PACTOLUS is an attempt to demonstrate that this can be achieved with digital simulation.

PACTOLUS includes all standard analog computer elements plus many special function blocks. The configuration and parameters are simply specified; runs may be interrupted and changes made at will. The output is recorded on the plotter during execution of the run. Variables of lesser interest are periodically recorded by the typewriter.

The author offers reprints of the FJCC paper upon request.

JOINT WESTERN AND MID-WESTERN REGION MEETING
1620 USERS GROUP
UNIVERSITY OF OKLAHOMA
NORMAN, OKLAHOMA
November 9, 10 and 11, 1964

STRAP WITH FORMAT

a

STEPWISE REGRESSION ANALYSIS PROGRAM

for the

1620 CARD OR TAPE SYSTEM

BY
W. WILCOXSON
AND
J. WOHLEVER*

U. S. NAVAL CIVIL ENGINEERING LABORATORY
PORT HUENEME, CALIFORNIA

*Now at Iona College, New Rochelle, New York.

INTRODUCTION

The reduction of data collected for evaluating an operation is one of the most difficult, time consuming and expensive problems encountered in the business, scientific and engineering fields. Except in simple cases, the data must be summarized in order to draw valid conclusions and predict future behavior with some confidence. Only after a laborious classification of the data, comparison of their relationships and finally discovery of a statement or formula that covers the range of the data may one expect to optimize an operation. Such an operation might be a scientific experiment, an engineering design, a process control or a business system.

When the relationship is of a quantitative nature, the appropriate statistical tool for the extraction of essential information from a collection of data and expressing it by a brief mathematical formula may be accomplished by means of a multiple curvilinear regression analysis. Sometimes this method is referred to as least squares, multiple correlation or partial regression analysis. This technique along with associated significance tests is a most valuable tool for evaluating operational data which is subject to a wide range of environmental error.

The use of this mathematical technique has increased greatly since the advent of modern computing systems. In general, it involves taking a set of observed values on a particular phenomenon and developing from them an empirical equation which may then be used for prediction of future values. It is especially useful in areas where little is known about the variables being examined and their interrelationships, or in areas where theoretically derived relationships between the variables are too complex to be handled conveniently. Regression analysis has been widely used by scientists and engineers to study correlations between observed data and to form equations relating one dependent variable with several independent variables. In recent years this tool has also become quite valuable in many non-technical areas such as business management and sales forecasting. Some applications of this technique at the U.S. Naval Civil Engineering Laboratory are given below.

A semi-empirical formula for the differential dose albedo for the backscattering of gamma rays on concrete is:

$$\alpha = \frac{0.0348 + 0.0224 \times 10^{26} K(\theta_s)}{1 + \cos\theta_o \sec\theta}$$

where the 0.2 Mev gammas are incident at an angle θ_o and are scattered at an angle θ through a scattering angle θ_s. $K(\theta_s)$ is the Klein-Nishina formula for the energy scattering cross section per electron.

After a one inch horizontal displacement, the holding power in Kips of a single deadman in sand is predicted as

$$P = 2.07 A^{0.863} e^{0.337 D}$$

where A is the facial contact area in square feet and D is the embedment depth to the top of the deadman in feet.

STRAP with FORMAT is a program that implements this technique on the 1620. A complete manual for using this program is given in this paper. The program is a major modification of STRAP[1]. None of the features or capabilities of STRAP have been sacrificed. It contains all the desirable features of SCRAP[3] and many features of the 7094 program BMD02R[4]. Capabilities not available in STRAP are listed below.

1. Variable format for input data.
2. Variable format for sequence numbers.
3. Force curve through origin.
4. Control card option to force all terms.
5. Solve $y = a_1 x_1$ and $y = a_o + a_1 x_1$ (not possible with SCRAP).
6. Weight observations.
7. Square root transformation (reduces the transformation time from about 600 milliseconds to 120 milliseconds).
8. Absolute value transformation.
9. Reverse sign transformation.
10. Extra variable read-in any time when data is in FORTRAN format.

11. Output raw sums, sums of squares and cross product (original matrix).

12. Output inverted matrix.

13. Output covariance matrix for coefficients.

14. Input the F level for entering a variable, the F level for removing a variable and the tolerance for entering a variable.

15. Tabulate observed versus predicted values.

16. Tabulate residuals.

17. Tabulate upper and lower confidence limits on the predicted values.

MATHEMATICAL DISCUSSION

Multiple curvilinear regression analysis is a method used to summarize data and to make inferences regarding the data. A best fit (in the sense of "least squares") is made of a set of observations on the independent and dependent variables entering into a given equation representing the variables. The equation must be linear in the parameters to be estimated. The equation is of the general form:

$$Y = b_0 + b_1 X_1 + b_2 X_2 + \cdots + b_m X_m \qquad (1)$$

where Y is the dependent variable; and $X_1, X_2, X_3, \ldots, X_m$ are the independent variables in the equation. The actual variables may appear non-linearly in the equation; e.g., if $x_1, x_2, x_3, \ldots, x_{m'}$ and y are the observed variables, then equation (1) could be

$$\log y = b_0 + b_1 x_1 + b_2 x_1^2 + b_3 \log x_2 + b_4 x_1 x_2 + \cdots + b_m f(x_1, x_2, \ldots, x_{m'})$$

where $f(x_1, x_2, \ldots, x_{m'})$ is some known function of the variables that does not involve the parameters to be fitted.

This program computes a sequence of multiple linear regression equations in a stepwise manner[2]. The procedure provides intermediate answers that are not available by normal computational techniques. The variable entered at each step is controlled by the intermediate results. The variable added to the equation is the one that makes the greatest reduction in the sum of squares of the residuals at that step.

Equivalently at that step is the variable which has the highest F value. Also at each step, the variables already in the equation are checked for significance. An insignificant variable will be removed before adding another variable. The sequence of intermediate regression equations fit to equation (1) might be:

$$Y = b_0 + b_5 X_5$$
$$Y = b_0' + b_2' X_2 + b_5' X_5$$
$$Y = b_0'' + b_2'' X_2 + b_5'' X_5 + b_7'' X_7$$
$$Y = b_0''' + b_2''' X_2 + b_7''' X_7 + b_9''' X_9$$
$$\vdots$$

In addition to the capability of providing intermediate answers, this program will produce the normal output required of a standard regression analysis program. The formulas for computing this standard output are given below. Here N is the number of observations, X_{ij} is value of the j^{th} transformed variable for the i^{th} observation, m is the number of transformed independent variables and w_i is the weighting factor for the i^{th} observation. For simplicity the value of the transformed dependent variable for the observation is denoted as $X_{i,m+1} = Y_i$. The predicted value for Y_i is given as $\hat{Y}_i$.

For the original matrix the sum of weights, raw sums and raw sums of squares are, respectively:

$$c_0 = \sum_{k=1}^{N} w_k$$
$$c_j = \sum_{k=1}^{N} X_{kj} w_k$$
$$c_{ij} = \sum_{k=1}^{N} X_{ki} X_{kj} w_k$$

Unless otherwise stated, in this and all the following formulas $i, j = 1, 2, 3, \ldots, m+1$. The means and residual sums are:

$$\bar{X}_i = c_i / c_0$$
$$a_{ij} = c_{ij} - c_i \bar{X}_j$$

If the curve is forced through the origin ($b_0 = 0$), then the residual sums are not computed and $a_{ij} = c_{ij}$; $i, j = 1, 2, 3, \ldots, m+1$.

Covariances, standard deviations and simple correlation coefficients are:

$$s_{ij} = \frac{a_{ij}}{N-1}$$

$$s_i = \sqrt{s_{ii}}$$

$$r_{ij} = \frac{a_{ij}}{a_{ii}\, a_{jj}}$$

Let $i_1, i_2, \ldots, i_K$ be the subscripts for the K variables in the equation after some step. If the curve is not forced through the origin, the pure constant is:

$$b_0 = \bar{Y} - \sum_{k=1}^{K} b_{i_k} \bar{X}_{i_k}$$

The standard error of the estimate is:

$$S = \sqrt{\frac{1}{N-p} \sum_{k=1}^{N} w_k (Y_k - \tilde{Y}_k)^2}$$

$$= \sqrt{\frac{1}{N-p} \left(a_{m+1,m+1} - \sum_{k=1}^{K} b_{i_k} a_{i_k m+1} \right)}$$

where $p=K$ if the curve is forced through the origin; otherwise, $p=K-1$. $Y_k - \tilde{Y}_k$ is the k^{th} residual, where if the curve goes through the origin the predicted value is:

$$\tilde{Y}_k = \sum_{j=1}^{K} b_{i_j} X_{k i_j}$$

and if the curve does not go through the origin

$$\tilde{Y}_k = \sum_{j=1}^{K} b_{i_j} X_{k i_j} + b_0$$

The multiple correlation coefficient is:

$$R^2 = \frac{1}{a_{m+1,m+1}} \sum_{k=1}^{K} b_{i_k} a_{i_k m+1}$$

The standard error of the j^{th} coefficient in the equation is:

$$S_{b_{i_j}} = S\sqrt{d_{i_j i_j}}$$

where $d_{i_j i_j}$ is a diagonal element in the inverted matrix. Its F value is:

$$F_j = \left(b_{i_j} / S_{b_{i_j}} \right)^2$$

Covariance of the estimated coefficients j and k in the equation is:

$$cov(b_{i_j}, b_{i_k}) = S^2 d_{i_j i_k}$$

where $d_{i_j i_k}$ is an element in the inverted matrix. Confidence limits for a predicted value of Y_m at some Student's t value for $N-p$ degrees of freedom are:

$$\tilde{Y}_m \pm tS\sqrt{1 + \frac{1}{N} + \sum_{k=1}^{K}\sum_{j=1}^{K} cov(b_{i_j}, b_{i_k})\left(X_{m i_k} - \bar{X}_{i_k}\right)\left(X_{m i_j} - \bar{X}_{i_j}\right)}$$

If the curve goes through the origin, use

$$\tilde{Y}_m \pm tS\sqrt{1 + \sum_{k=1}^{K}\sum_{j=1}^{K} cov(b_{i_j}, b_{i_k}) X_{m i_k} X_{m i_j}}$$

INPUT

The input data consist of two decks. The first deck, called the definition deck, specifies the size of the problem and the transformations to be performed on the observations. The second deck, called the data deck, contains the observations. Both of these decks are placed between the program deck for part 2. If the option for giving the names of the variables in the output is selected, then a third deck containing these names must be placed between the program decks for part 2 and part 3. If a tabulation of the transformed dependent variable versus its predicted value, the residuals, and the confidence limits on the predicted value, then the transformed data deck produced by part 1 must follow program deck 4.

Definition Deck

This deck must contain the following cards (Format Card is optional) and must be in the order given.

Title Card
Problem Definition Card
F-Card
Disregarded Observations Card
Transformation Constant Card(s)
Transformation Code Card(s)
Format Card (optional)

Preparation of each type of card of the definition deck is listed below.

Title Card: May contain up to 80 columns of alphanumeric information which will be used as a heading for the program output.

Problem Definition Card: Defines the size of the problem and specifies the information to be in the output. The following card columns are used for this purpose. Each number must be right hand justified in its field. All data are numeric. (A blank is treated the same as zero.)

1-2 Number of independent variables to be read in with each observation (1 to 40)

3-4 Number of dependent variables to be read in with each observation (1 to 14)

5-6 Number of transformed independent variables to be considered in the solution. (1 to 39 if input data and transformed data output are both in excess fifty format; otherwise, 1 to 33.)

7-8 The dependent variable to be considered in the solution (1 to 14)

9-10 The variable to use as the weighting factor. If left blank, each observation is weighted equally.

11-12 Number of constants (0 to 20)

14 All terms are forced if a non-zero digit is punched in this column.

16 Curve is forced through origin if a non-zero digit is punched in this column.

18 Only the last step in the development of the regression equation output if a non-zero digit is punched in this column.

20 Input data is in the standard FORTRAN notation as output by STRAP if a non-zero digit is punched in this column (sequence numbers are checked).

22 Input data is in the standard excess fifty floating point format as output by STRAP if a non-zero digit is punched in this column.

24 Names of the variables are output if a non-zero digit is punched in this column. (Variable Name Deck is required as part of the input.)

26 Transformed data will be output in STRAP's standard excess fifty folating point format if a non-zero digit is punched in this column.

28 Original matrix (raw sums, sums of squares and cross products) is output if a non-zero digit is punched in this column.

30 Residual sums of squares and cross products are output if a non-zero digit is punched in this column.

32 Mean values and standard deviations are output if a non-zero digit is punched in this column.

34 Simple correlation coefficients are output if a non-zero digit is punched in this column.

36 Inverted matrix is output if a non-zero digit is punched in this column.

38 Variance-covariance matrix for regression coefficients is output if a non-zero digit is punched in this column.

40 The transformed dependent variable and its predicted value, residuals, and upper and lower confidence limits on the predicted value if a non-zero digit is punched in this column.

42 If a non-zero digit is punched in this column, Student's t for the desired probability level on the upper and lower confidence limits is entered from the typewriter during program 4; otherwise, a 95% confidence level is used.

F-Card: F-level for entering a variable, F-level for removing a variable and tolerance level for entering a variable in FORMAT(3E15.8)

If the F-Card is blank, these will be set to 0.005, 0.01 and 0.0001, respectively, provided all terms are not forced. When all terms are forced, the values are automatically set to sero regardless of the data on the F-Card. The tolerance level[2] is a control on the size of the diagonal elements and reduces the possibility of degenacy when an "independent" variable is approximately a linear combination of the other dependent variables.

Disregarded Observation Card: Denotes the observation numbers not to be included in the regression analysis. Rather than remove these observations from the data deck, they may be eliminated from consideration by including their three digit observation numbers in ascending order starting in column one of this card. If leading zeros are not punched, a number must be right-hand justified in its field. A maximum of twenty observations may be disregarded in this manner. If no observations are to be thrown out, a blank card must be inserted. If the input data is in excess fifty notation, then the observation number is the same as the number punched in column 78 to 80 of the data deck. When the input data is in FORTRAN notation, the first set of values for the independent and dependent variables is counted as observation number 1, the second as observation number 2, etc. (Disregarded observations will not be in the transformed data deck.)

Transfromation Constant Card(s): In applying transformations to the input data, constants may be required. Up to twenty constants can be used. These must be entered five per card in FORMAT(5E15.8) with a maximum of four cards. A blank card must be inserted if no constants are used. The first constant will be considered by the transformation routine as constant number 1, the second as number 2, etc.

Transformation Code Card(s): The transformation operations to be performed on the observations are defined using the 10 digit codes described below. These codes are entered, 8 per card, with a maximum of 5 cards or 40 transformations. The last transformation code must be all zeros or blanks to signal the end of the transformations. If no transformations are to be used, a blank card must be inserted. Each transformation code is in the format TTIIJJCCKK where:

TT is the transformation to be performed

II is the number of the variable involved

JJ is the number of the second variable involved (if necessary)

CC is the number of the constant to be used (if necessary)

KK is the number of the transformed variable

Zeros or blanks must be entered for elements of the code that are not relevant. Each number in the code is right-hand justified in its two-digit field. Variable numbers from 1 to 39 refer to the 1st through the 39th independent variable, variable numbers from 41 to 54 refer to the 1st through the 14th dependent variable and constant numbers from 1 to 20 refer to the 1st through the 20th constant.

Transformation available:

TT Code	Name	Operation
Blank	End	End of transformations
1	Add	$X_{II} + X_{JJ}$ (or C_{CC}) $\rightarrow X_{KK}$
2	Subtract	X_{JJ} (or C_{CC}) $- X_{II} \rightarrow X_{KK}$
3	Multiply	$X_{II} * X_{JJ}$ (or C_{CC}) $\rightarrow X_{KK}$
4	Divide 1	X_{JJ} (or C_{CC}) $/ X_{II} \rightarrow X_{KK}$
5	Divide 2	X_{II} / X_{JJ} (or C_{CC}) $\rightarrow X_{KK}$
6	Logarithm (base 10)	$\text{Log}(X_{II}) \rightarrow X_{KK}$
7	Logarithm (base e)	$\text{Ln}(X_{II}) \rightarrow X_{KK}$
8	Exponential (base 10)	$10^{X_{II}} \rightarrow X_{KK}$
9	Exponential (base e)	$e^{X_{II}} \rightarrow X_{KK}$
10	Exponential (variable base)	$X_{II}^{C_{CC}} \rightarrow X_{KK}$
11	Sine	$\text{Sin}(X_{II}) \rightarrow X_{KK}$
12	Cosine	$\text{Cos}(X_{II}) \rightarrow X_{KK}$
13	Summation	$\sum_{i=II}^{JJ} X_i \rightarrow X_{KK}$
14	Replacement	$X_{II} \rightarrow X_{KK}$
15	Test	If X_{II} is negative, go to transformation number JJ If X_{II} is positive, go to transformation number KK If X_{II} is zero, go to the next transformation
16	Square Root	$\sqrt{X_{II}} \rightarrow X_{KK}$

17	Absolute Value	ABS $(X_{II}) \longrightarrow X_{KK}$
18	Reverse Sign	$-X_{II} \longrightarrow X_{KK}$

This flexible transformation system provides the ability to perform a number of successive operations on the same variable; thus it is possible to use any polynomial or exponential expression as a variable. For example, the sequence of codes to build a variable of the form:

$$\frac{(X_3^2 + X_2 + C_2) \ / \ X_5}{X_1} = X_{10}$$

is as follows(The transformation numbers are based on the assumption that these are the first transformations for the problem):

Transformation Number	Code	Operation
1	0303030010	$X_3 \times X_3 \longrightarrow X_{10}$
2	0110020010	$X_{10} + X_2 \longrightarrow X_{10}$
3	0110000210	$X_{10} + C_2 \longrightarrow X_{10}$
4	0510050010	$X_{10}/X_5 \longrightarrow X_{10}$
5	0601000039	$\log X_1 \longrightarrow X_{39}$
6	0310390010	$X_{10} \times X_{39} \longrightarrow X_{10}$
7	0810000010	$10^{X10} \longrightarrow X_{10}$

Leading zeros and zero fields (each code consists of five 2-digit fields) may be replaced by blanks. Variable 39 was used as temporary storage and should be out of range of the number of independent variables read-in with each observation. The independent variables read-in with each observation will be automatically stored at independent variable number 1,2,3,.... Similarly, the dependent variables will be stored at dependent variable number 41,42,43,.... The transformed independent variables to be considered in the regression analysis must be stored at independent variable number 1 through the number specified in column 5 and 6 of the definition card by the end of the transformations on the data for each observation.

Format Card (Optional): If a digit is <u>not</u> punched in either column 20 or 22 of the Definition Card, then a Format Card is required, and the input data deck must correspond to the specifications on the Format Card. Mixing of FORTRAN type and excess fifty floating point data is not permitted.

The Format Card permits a flexible data input. With some exceptions, the Format Card is compiled the same as a FORMAT statement in PDQ FORTRAN. An "H" or Hollerith specification may <u>not</u> be used (an "X" may be used.) The I type specification is a notation for checking sequence. A statement number is not required and must not be used. Use of the word "FORMAT" is optional; however, a left parenthesis is required. Card column 73 through 80 are not considered by the compiler. Imbedded blanks are ignored. Should a set of specifications require more characters than are available on the Format Card, then the set may be continued on the next card beginning in column 7. A digit must be punched in column 6 of the continuation card, and the first 5 columns must not be used. A specification may not be split between cards. Any Format Card to be followed by a continuation card should terminate with a comma. ERROR 5 will occur if the set of specifications exceeds storage capacity. As in PDQ FORTRAN, the first right parenthesis encountered terminates compilation; therefore, any additional right parenthesis that are required to mate left parenthesis must be on the same card. Mating of parenthesis is not mandatory.

Permissible format specifications* are I type (Iw) Ftype (Fw•d), E type (Ew•d), X Type (wX) and slash (/). All the numeric field specifications may be repeated by preceding it by the number of required repetitions, such as (3E9.2) which is equivalent to (E9.2, E9.2, E9.2). Data in E type notation may be input under the F type specification. The converse is also true, i.e., both the E and F specification are treated exactly the same during input data conversion under PDQ FORTRAN.

If the input list <u>is not</u> satisfied when <u>the</u> "right parenthesis" is encountered at the time of data conversion, a new record is read, and the specifications would be as if repeated from the last left parenthesis. For example, (3F9.2,5X / F10.4,I5, (E15.8,I3)) would be the same as (3F9.2,5X / F10.4, I5,E15.8,I3 / E15.8, I3 / E15.8, I3 / .... Note that the right parenthesis acts as a slash and not the left parenthesis. No specification beyond the first right parenthesis is compiled because of the "infinite repeater" nature of the right parenthesis in FORTRAN I. (More than two left parenthesis is of no advantage to the user.)

If the input list <u>is</u> satisfied before the "right parenthesis" is encountered at the time of data conversion, the remaining specifications are examined to determine if any are I type. E and F type will be treated as X type, e.g., E10.3 would be considered as 10X during the examination. If an I type is discovered, sequencing <u>will be</u> checked. The investigation of the remaining specifications will be terminated if either a slash or the right parenthesis is encountered. Scanning of the remaining format statement is not done in FORTRAN

*Except for I type which is restricted to sequence checking, all the specifications are defined in PDQ FORTRAN (1620 Users Group Number 2.0.031)

The E and F type specifications are used to describe the format of the data for the independent and dependent variables. An X specification is used to skip W-characters in the input record (card). A slash forces reading the next record. The I type specification defines the format of the sequence numbers. Sequencing must be in fixed point notation (maximum of 4 digits with truncation of excessive high order digits). No restriction is placed on the field width, w, or the sign. The sequence number is not available as variable data. (For additional information see Special Features.)

Data Deck

The data deck may be in one of the three formats acceptable to STRAP. It may be in the standard FORTRAN notation as output by STRAP, the standard excess fifty floating point notation as output by STRAP, or in the FORTRAN notation as specified by the Format Card. A description of the standard transformed data output format is given under OUTPUT. In all cases, the first piece of variable data is taken as the X_1, the second as X_2, the third as X_3, etc., until the independent variable list is satisfied. Then the next piece of variable data is accepted as $Y_1 = X_{41}$, the next as $Y_2 = X_{42}$, the next as $Y_3 = X_{43}$, etc., until the dependent variable list is satisfied. (Note that the replacement operation, transformation code 14, permits interchanging the so called "independent" and "dependent" input variables.)

In no case are the sequence numbers available as variable data. Sequence of standard Data Decks are always checked. In non-standard Data Decks, sequence* is checked only if requested. In standard decks sequence numbers are always in the right most columns; whereas, in non-standard decks sequence numbers, if any, may be interspersed in the variable data for each set. (See Special Features for a description of sequence checking for non-standard decks.)

The first card of program deck 2 is an "end of Data Deck" control card. (Column 1 of this card contains a record mark for this purpose.) That is, the user need not count the number of observations because the counting is done by the program and is part of the output. The arbitrary control information necessary to stop the counting operation is supplied by the program — not the user.

Variable Name Deck** (optional)

If the option to give the names of the variables in the output is selected, then this deck must be placed between the program decks for part 2 and part 3. One card must be used for the name of each independent

*Not a true sequence check. It is not necessary that the numbers be sequential, only in an ascending order.

** Variable names may be output by STRAP with FORMAT without requiring the output of values and standard deviations. Also the names are automatically numbered in the output for reference. These features are not available under STRAP, 6.0.066.

variable considered in the solution, and one for the name of the dependent variable. (Only the first 35 characters on each card will be in the output.) These cards must be arranged in ascending order with the dependent variable name last.

Transformed Data Deck (optional)

If the option to tabulate the transformed dependent variable versus its predicted value, the residuals and confidence limits on the predicted value, then this deck must follow program deck 4. If no transformations are made and the data is punched in either of the standard formats, the original data deck may be used; otherwise, the transformed data must be output (Sense Switch 1 on) by program 1 for input to program 4.

OUTPUT

The output from the program has been made quite flexible in that most portions of it can be suppressed by console switch or control card option. The following section describes these outputs in the order in which they appear in each part of the program and indicates the method for altering or suppressing certain of these.

Typeout of Transformed Variables

The transformed independent variables to be considered in the solution and all the dependent variables will be typed out in order when console switch 1 is on. The observation number will be typed first followed by the independent variables and the the dependent variables separated by a return carriage on the typewriter. The variable data will be output in excess fifty notation if requested on the problem Definition Card; otherwise, output is by FORMAT(5E15.8).

Punching of Transformed Variables

The transformed independent variables to be considered in the solution and all the dependent variables will be punched out in the standard data card format when console switch 2 is on. These may then be used as data cards for future regression runs, eliminating the need for transforming the input variables. (This also operates in the same fashion with the paper tape version of the program when console switch 4 is on. The output tape produced may be used as an input data tape.)

Standard FORTRAN Data Deck: Each observation is contained on two sets of cards, one set for independent variables and one for the dependent variables. The independent variables are punched 5 per card by FORMAT(5E15.8,I5), where the I specification is for output of the card number. The dependent variable* is punched by FORMAT(E15.8,60X,I5).

*Only the transformed dependent variable entering the regression analysis is included in the Transformed Data Deck. The standard Input Data Deck is not restricted to one dependent variable.

Standard Excess Fifty Data Deck: Each observation is contained on two sets of cards, one set for independent variables and one for the dependent variable*. The variables are entered on the first 70 card columns in excess-fifty floating point notation (7 variables/card). A negative number is indicated by an X punch over the low order digit of the number. The last 9 card columns are used for identification purposes as follows:

72-75	Regression Number, to identify the data deck.
76	0 indicates independent variables 1 indicates dependent variables
77	Card Number for each set. Independent variables may go from 1 through 6, dependents from 1 through 2.
78-80	Observation Number.

Solution Output

The solution is normally punched out on cards for listing by a 407 with an 80-80 board. It may also be typed out in the same format by putting console switch 3 on. (The paper tape version of the program, using console switch 4 on, automatically types the output.)

Each reference to a variable in the output is by a two digit number. For example, if the transformed independent variables are X_1, X_2, X_3, and X_4, then these are represented by 01, 02, 03, and 04. In this example, a reference to the dependent variable is by the code 05.

Elements in arrays are referenced in the output by a four digit code representing two variable codes. In the above example, 0105 would be a joint reference to the independent variable X_1 and to the dependent variable $Y = X_5$. If the original matrix is output, the raw sums are identified by the codes 0001, 0002, etc. The code 0000 identifies the sum of the weights (same as the number of observations if equal weights).

Order of the output is:

	Description	Optional	Program Deck
1	Transformed variables	yes	1
2	Title card	no	2
3	No. of Independent Variables	no	2
4	Dependent Variable No.	no	2
5	Raw sums (original matrix)	yes	2
6	Residual sums	yes	2
7a	Names of Variables	yes	2
7b	Names, mean values and Standard Deviations	yes	2
8	Simple Correlation Coefficients	yes	2
9	Stepwise output: a. Step number b. Entering or leaving variable number c. Pure constant d. Standard error of the estimate e. F-value for entering or leaving variable f. Regression coefficients g. Standard error of each coefficient h. Multiple correlation coefficient (R^2)	yes	3
10	Final step	no	3
11	Inverted matrix	yes	4
12	Covariance matrix	yes	4
13	Tabulations: a. Predicted versus observed b. Residuals c. Upper and lower confidence limits	yes	4

*Only the transformed dependent variable entering the regression analysis is included in the Transformed Data Deck. The standard Input Data Deck is not restricted to one dependent variable.

SPECIAL FEATURES

Paper Tape Operation

The STRAP program for the 1620 card system can be operated as a paper tape program with console switch 4 on. The card decks for Part1, 2, 3 and 4 must be converted to card images on paper tape by means of a card to tape converter. All 80 columns must be punched with an end of line punch in column 81 on tape. The preparation of input data is identical to that described for the card program except that a definition tape, a data tape and possibly a variable name tape must be punched from those cards in the same manner as described above for the program cards. The operating instructions are as follows:

1. Clear Memory
2. Thread and Load Program Part 1 (360000000300)
3. Thread Definition Tape when message types out then press start
4. Thread Data Tape when message types out, then press start
5. Thread Part 2 when message types out, then press start

6. Thread variable name tape if message types out, then press start

7. Thread Part 3 when message types out, then press start

8. Thread Part 4 if required when message types out, then press start

9. Thread Transformed Data Tape if message types out, then press start.

Extra Card Read-In

STRAP w/FORMAT can handle up to 40 independent variables and up to 14 dependent variables in the Input Data Deck. In most cases there will be unused data area, that is, most problems will not use as many as 54 variables. These unused variables are not affected by the reading of data and therefore can be put to use to store constants or as areas for temporary storage to allow shifting of variables from one observation to the next. In order to initialize these unused variables to some desired value or to change their value during the processing of observations, a method has been worked out to allow more than the normal number of variables to be read in with certain observations. The procedure for Extra Card Read-In is different for standard excess fifty data and data in FORTRAN notation. For Extra Card Read-In, there is no distinction between standard FORTRAN data and data specified by a Format Card.

For data in FORTRAN notation, an Extra Card Read-In is indicated by an Asterisk Card. Asterisk Cards may be placed anywhere in the Data Deck. Each card has an asterisk punch in column 1. In column 2 and 3 is the variable number desiginating the unused storage area where the number (by E15.8 format) in column 6 through 20 will be stored. Once stored, the data may be transformed or manipulated the same as any normal input variable.

For standard excess fifty data, an Extra Card Read-In is indicated by a non-zero punch in column 71 of what is normally the last independent variable card or last dependent variable card. The input routines for handling excess fifty data are radically different from those for handling FORTRAN data. Since as many as seven numbers may be punched in each card in excess fifty notation, the routine stores all seven sequently even though some may not be variable data; therefore, if there are unused fields on the last independent variable card, one has an extra "variable" capability that is not available with FORTRAN Data. If this capability is not sufficient, then the Extra Card Read-In may be used as many times as necessary - providing there are no more than 6 cards (40 variables*)

* The 41st and 42nd variables on the 6th card are replaced by the 1st and 2nd sependent variables.

with each set of independent variables or 2 cards (14 variables) with each set of dependent variables.

Sequence Checking for Non-Standard Input Data Decks

When a data deck must be handled many times, verification that it is in order* is standard operating procedure. Standard input data decks to STRAP w/FORMAT are always checked for order; however, checking is optional for non-standard decks. The option results from not being able to force the user to punch an I type specification in the Format Card. Similarly there is no restriction on the location of the sequence field on the data cards. And within the limits of the card, no restriction on the number of sequence fields per card. The only restrictions are that the number must be 4 or less digits (otherwise, excessive high order digits are truncated before comparison), and that the numbers be in an ascending order (not necessarily sequential).

Typewriter Input of Student's t

If a 95% confidence level is not wanted, Student's t at the desired probability level for the upper and lower confidence limits may be entered from the typewriter during execution of program 4. When the option to input Student's t is indicated on the problem Definition Card, the number of degrees of freedom (number of observations less the number of parameters fitted) will be typed out followed by a request for Student's t. The desired t value is entered in E15.8 format. After entry press the Release and Start buttons on the console. If an error is made in typing, turn on Sense Switch 4 and then press Release and Start. (If operating on a paper tape machine, Switch 4 must be turned off before typing Student's t and turned back on after pressing Release and Start.) When another request is made for Student's t, turn off Switch 4 and precede as before. For example, if there are 27 observations and 5 parameters entered into the last step of the regression analysis, the following would be typed out.

DF 22

T

*To obtain meaningfull results using STRAP, the order of the variables must not be changed from one observation to the next. In an algebraic sense, the results obtained by STRAP are independent of the order of the observation; however, the finite computational process may produce different results if the observations are rearranged.

Assume that a 99% confidence level is required. Then the typewriter listing after entering t would appear as

DF 22

T 2.819

OPERATING PROCEDURE

Order of Input Decks

Assemble program and input decks as follows:

Program Deck 1

Definition Deck

Data Deck

Program Deck 2

Variable Name Deck (if necessary)

Program Deck 3

Program Deck 4 (if necessary)

Program Switch Options

Set the console switches as required:

Sw 1 on to punch out transformed data

Sw 2 on to type out transformed data

Sw 3 on to type output instead of punching it

Sw 4 on if program is to be run as a paper tape program (See Special Features)

Loading of Assembled Decks

The sequence of steps required to use STRAP are detailed below. Restart procedures in event of an error during processing are given under the section titled Error Messages.

1. Press Reset on the console
2. Set program switches

3. Place assembled decks in the read hopper
4. Clear punch
5. Place blank cards in the punch hopper
6. Press Punch Start
7. Press Reader Lad. The message "SIGN LOG" will be typed, core will be cleared and program 1 will be loaded followed by the message "LINEAR STEPWISE REGRESSION ANALYSIS."
8. After the data deck is processed, if the message "REMOVE TRANSFORMED DATA FROM PUNCH" is typed, clear the punch. Date and label the deck TRANSFORMED DATA FOR "title of the regression problem".

   If the option to tabulate predicted versus observed, residuals and confidence limits on the predicted values is requested, place the Transformed Data Deck on top of program 4 in the reader. Press Punch Start and Start on the console to continue.
9. Except for possible error stops, processing will continue to the end without further interruptions. At the end of processing (machine is in manual mode), clear the punch hopper. Date and label the deck REGRESSION ANALYSIS FOR "title of the regression problem".
10. List the output on 407 with an 80-80 board.

Error Messages

When an error occurs in a floating point subroutine, an error message of the form RRRRROOSS will be typed out, where R is the return address to the main program and S is a code which identifies the special error condition. The conditions that can arise in the floating point subroutines and the error code associated with each are:

01	FA, FS	Characteristic Overflow
02	FA, FS	Characteristic Underflow
03	FM	Characteristic Overflow
04	FM	Characteristic Underflow
05	FD	Characteristic Overflow
06	FD	Characteristic Underflow Course of action depends upon the digit at location 401.
07	FD	Attempt to divide by a floating point number with a zero mantissa. May not continue execution of the subroutine

08	FSQR	Attempt to take the square root of a negative number. May press the start key to continue execution of the subroutine, finding the square root of the absolute value of the number, or may branch back to the main program.
09	FSIN, FCOS	Input argument has a characteristic greater than 58. May not continue execution of the subroutine as all significance would be lost in the result.
10	FSIN, FCOS	Input argument has a characteristic greater than or equal to 52 or less than or equal to 58. May continue execution of the subroutine with some loss of significance in the result.
11	FEX FEXT	Characteristic overflow
12	FEX FEXT	Characteristic underflow Course of action depends on digit at location 401
13	FLN FLOG	Input argument has a zero mantissa. May not continue execution of the subroutine
14	FLN FLOG	Input argument is negative. May continue to execute the subroutine, computing with the absolute value of the number

If the data deck is not punched or assembled properly, the message DATA OUT OF ORDER will be typed out and the program will halt. To resume processing, remove the last two cards from the read stacker and determine which observation number was being worked on. Remove the cards from the read hopper and non-process run out cards into the read select stacker. Correct the error condition and then reload the cards into the read hopper starting with the first card of the observation which was in error. Press Reader Start to continue with the processing of observations.

In converting FORTRAN type data to excess fifty type, the message ERROR IN INPUT DATA will be typed out if the data is outside the allowable range. If the number is too large, it will be replaced with $.99999999 \times 10^{49}$. If the number is less than $.99999999 \times 10^{-49}$ in absolute value, it will be replaced with zero. Processing is not stopped under either condition.

When an error is detected when compiling a Format Card, the message FORMAT ERROR followed by the error number will be typed out. The errors detected are:

1 The first non-blank character is not a left parenthesis, or the first non-blank character following the word FORMAT is not a left parenthesis.

2 A record width of greater than 80 characters is specified.

3 More than two digits are used to specify the field width, the number of decimal digits or the number of iterations for an E or F type specification.

4 (a) Special character (= @ – * + . ,) in a numerical field specification.
(b) Alphabetic character other than E or F in a numerical field specification.
(c) Decimal point missing in an E or F type specification.
(d) The number of decimal digits has not been given in an E or F type specification.
(e) A single specification is split between cards.
(f) A record mark appears in a numerical field specification.

5 The set of format specifications exceeds storage capacity.

Except for number 5, when an error occurs during the compilation of a format statement, remove the cards from the read hopper and non-process run out cards into the read select stacker. Correct the error condition and then reload the cards in the read hopper starting with the first Format Card. Press Reader Start then Start to restart the compilation.

REFERENCES

1. Colville, A.R., and L.S. Holmes, "STRAP, A Stepwise Regression Analysis Program for the 1620 Card or Tape System," IBM 1620 Users Group, White Plains, New York (Write-up number 6.0.066 dated 15 January 1962).

2. Efroymson, M.A., "Multiple Regression Analysis", Mathematical Methods for Digital Computers, Part V(17), ed. A. Ralston and H.S. Wilf, John Wiley and Sons Inc., New York (1960), pp191-203.

3. Leeson, D.N.,"SCRAP / Sixteen-Twenty Card Regression Analysis Program", IBM 1602 Users Group, White Plains, New York (Write-up number 6.0.003 dated 1962).

4. Dixon, W.J., ed., "BMD02R Stepwise Regression," BMD Biomedical Computer Programs, Health Science Computing Facility, Department of Preventive Medicine and Public Health School of Medicine, University of California at Los Angeles (January 1, 1964) pp233-247.

SOLUTION TO NETWORK EQUATIONS

USING TOPOLOGICAL FORMULATION

By

A. F. ATKINS

and

R. H. SEACAT

November, 1964

Engineering Research Laboratories

Texas Technological College - Lubbock

SOLUTION TO NETWORK EQUATIONS USING TOPOLOGICAL FORMULATION

INTRODUCTION

One of the problems in the analysis of electric networks by the use of digital computers is the time required to calculate the number of trees in a network and the storage space required to store the results. Several investigators have developed programs for analyzing networks using the method first postulated by Kirchhoff. In this paper, we shall use a method developed by Percival and improved upon by investigators in circuit theory at Texas Technological College to obtain a program for analyzing electric networks containing independent sources and linear, passive circuit elements.

A resume of Percival's method and the attendant topological formulae will be discussed first. The program will be developed for the method described and examples will be used to illustrate the programming techniques.

A brief review of the methods of a network solution due to W. S. Percival[1] and R. H. Seacat[2] will be given. Both of these methods have advantages over the normal determinant techniques and several of these advantages are:

1. Network equilibrium equations are not required.
2. Their solution does not contain redundant terms. It is a well-known fact that the negative terms obtained during the expansion of a passive network determinant must have a like positive term since all coefficients of the expanded determinant must be positive; therefore, considerable work is saved by using a method that eliminates this redundancy.
3. There is less chance for error when using these methods instead of other approaches to the solution of the network equations. This is true because of the relatively simple rules used in conjunction with these two methods and the many intermediate checks available during the solution.

A BRIEF REVIEW OF PERCIVAL'S METHOD AND THE METHOD OF RESIDUAL NETWORKS AS IT APPLIES TO PASSIVE NETWORKS

If the electrical network is to be solved on a nodal basis, its equilibrium equations can be put into the form:

$$[I] = [Y][V] \tag{1}$$

where [I] is a column matrix with N rows, its entries being known values of independent current sources; [Y] is the network admittance matrice with N rows and columns, and [V] is a column matrice with N rows. Its entries are the unknown voltages across the N node pairs.

Let the determinant value of [Y] be denoted by:

$$\Delta = |Y| = \Sigma \text{ tree edge admittance product over all trees.} \tag{2}$$

The last part of Equation 2 will be used extensively in applying Percival's method. To illustrate this point further, consider the simple network shown in Figure 1.

Since admittances in parallel add, the simplified graph of Figure 2 will be used to calculate the expanded value of the admittance matrix of Figure 1.

In Figure 2, let the numbers 1, 2, 3, 4, and 5 be defined as:

$$\begin{aligned} 1 &= G_1 + SC_1 \\ 2 &= G_2 \\ 3 &= G_3 + SC_3 \\ 4 &= G_4 \\ 5 &= G_5 + SC_5 \end{aligned} \tag{3}$$

where S is the operator $\frac{d}{dt}$.

Using Percival's "network equations" one can decompose the network as shown in Figure 2. This figure indicates that the trees of the graph to

the left of the equal sign are equal to the trees of the graphs on the right. Note that edge 3 is removed in the first graph on the right of Figure 2; it is then replaced by a short-circuit, causing edges 1 and 2 to be in parallel in one part of the graph, 4 and 5 to be paralleled in another section of the graph. Since the trees of any separable network are simply the product of the trees of each sub-network, one can write at once, with the use of Figure 2.

$$= 124 + 245 + 451 + 512 + 3(1 + 2)(4 + 5) \tag{4}$$

From the very definition of a tree it is obvious that all trees, from the same graph, contain exactly the same number of edges and this number must be one less than the number of vertices. This is very useful in checking to see if some careless error has been made while calculating .

If Equations 3 and 4 are combined one has the following results:

$$\begin{aligned} \Delta = {} & C_1C_3C_5S^3 + [C_1C_3G_5 + C_1C_3G_4 + C_3C_5G_2 + C_3C_5G_1 + C_1C_5G_4 \\ & + C_1C_5G_2 + C_1C_5G_3]S^2 + [G_2G_5C_3 + G_2G_4C_3 + G_1G_5C_3 + G_1G_4C_3 \\ & + G_3G_5C_1 + G_3G_4C_1 + G_2G_3C_5 + G_1G_3C_5 + G_1G_2C_5 + G_2G_5C_1 \\ & + G_1G_4C_5 + G_4G_5C_1 + G_2G_4C_5 + G_2G_4C_1]S + [G_1G_2G_4 + G_2G_4G_5 \\ & + G_1G_4G_5 + G_1G_2G_5 + G_1G_3G_4 + G_1G_3G_5 + G_2G_3G_4 + G_2G_3G_5]. \end{aligned} \tag{5}$$

Note that Equation 5 contains thirty terms. If the equilibrium equation had been written and the determinant of the admittance matrix had been expanded, one would have gotten forty-two terms, twelve of which are redundant.

The Z parameters for this network will now be found. Also, voltages at various points will be calculated, assuming the network to be driven at both ports by current sources.

Let

$$Z_{11} = \Delta_{11}/\Delta, \tag{6}$$

$$Z_{12} = Z_{21} = \Delta_{12}/\Delta \text{ (for passive network)}, \tag{7}$$

and

$$Z_{22} = \Delta_{22}/\Delta, \tag{8}$$

where Δ_{11} is found by first placing a short circuit across the input terminals and then proceeding in exactly the same manner as if the Δ of the remaining graph was desired. Δ_{22} is found through the identical process except the short circuit is placed across the output terminals instead of the input. To calculate Δ_{12}, refer to Figure 3. A path from the terminal marked 1 and 2 is noted at the same time another non-touching path is chosen from terminals 1' and 2'. A product is made with all edge admittances used in the paths and with the trees remaining in the graph after all vertices used in the paths are shorted. This process is repeated for all possible combinations of paths that do not touch. The results obtained from this process are then summed over all paths. Next refer to Figure 3b. The same technique as outlined above is followed except the paths are crossed as shown. After summing over all paths, this result is subtracted from the first sum. This difference is Δ_{12}. Note that in all ladder type networks, there can never be any non-touching crosspaths.

Let us place a short across the input terminals of the network of Figure 1. This network has the graph shown in Figure 4a. Figure 4b is the remaining graph after the output terminals are shorted. Therefore, using the graphs of Figure 4, one obtains directly:

$$\Delta_{11} = (G_2 + G_3 + SC_3)(G_4 + G_5 + SC_5) + G_4(G_5 + SC_5) \tag{9}$$

and

$$\Delta_{22} = (G_1 + G_2 + SC_1)(G_3 + G_4 + SC_3) + G_2(G_1 + SC_1) \tag{10}$$

Since there is only one path from input to output meeting the restrictions previously mentioned,

$$\Delta_{12} = G_2G_4. \tag{11}$$

Substituting Equations 9, 10, and 11 into Equations 6, 7, and 8 gives the Z parameters of the network.

Next suppose Figure 1 has a current source of magnitude I_1 and I_2 connected at terminals 1-1' and 2-2' respectively. Superposition can be used to determine the voltage at any terminal pair within the network using techniques already learned from Percival. For example, $V_{2\text{-}2'}$ is given by the expression:

$$V_{2\text{-}2'} = Z_{22}I_2 + Z_{12}I_1. \tag{12}$$

If the voltage is desired at some other point, say across terminals AB, as shown in Figure 1, one simply thinks of terminals AB as the output of the circuit, driven first at terminals 1-1', and then at terminals 2-2'. The voltage V_{AB} is then

$$V_{AB} = V_{AB1} + V_{AB2} \tag{13}$$

where

$$V_{AB1} = I_1 \frac{(G_2)[G_4 + G_5 + SC_5]}{\Delta}; \tag{14}$$

and

$$V_{AB2} = \frac{I_2(G_4)[G_1 + G_2 + SC_1]}{\Delta}. \tag{15}$$

A residual network is defined as one in which all of the storage elements are set to extreme values of zero or infinity in a given order.

For example, if there is a single capacitor in a network with the remaining elements being resistors, the principal determinant may be expanded in the form:

$$\Delta = SC_1\Delta_1 + \Delta_1', \tag{16}$$

The determinant with the unprimed subscript denotes the element with that subscript has been set to the extreme value of infinity. Further the determinant with the primed subscript represents a network with the element set to a value of zero. In Equation 16, this is tentamount to shorting a capacitor in the network represented by Δ_1 and open circuiting the capacitor represented by $\Delta_{1'}$. If the network contains an inductance, L_2 and a capacitance C_1, it is seen that:

$$\Delta = SC_1\left(\frac{1}{SL_2}\Delta_{12} + \Delta_{12'}\right) + \frac{1}{SL_2}\Delta_{1'2} + \Delta_{1'2}$$

or

$$\Delta = S^2L_2C_1\Delta_{12'} + S(C_1\Delta_{12} + L_2\Delta_{1'2'}) + \Delta_{1'2} \qquad (17)$$

For a three storage element network, it may be seen that

$$\Delta = S^3L_2C_1C_3\Delta_{12'3} + S^2(L_2C_1\Delta_{12'3'} + C_1C_3\Delta_{123'} + L_2C_3\Delta_{1'2'3'}) + S(C_1\Delta_{123'} + L_2\Delta_{1'2'3'} + C_3\Delta_{1'23}) + \Delta_{1'23'} \qquad (18)$$

From an observation of Equations 16, 17, and 18, it is seen that the number of terms in each coefficient of S^K forms a Pascal triangle as shown below:

K = 1			1		1	
K = 2		1		2		1
K = 3	1		3		3	1
K = 4	1	4		6	4	1
K = 5	1	5	10	10	5	1

Each term is arrived at by multiplying the determinant of a selected residual network by the values of the elements set to extreme value of infinity. For example, the determinant $\Delta_{1'2'3'}$ in Equation 18 represents a network of resistors derived from the original network by setting the elements C_1 and C_3 to the extreme values of zero and the inductance L_2 to infinity.

The term must have the value $L_2\Delta_{1'2'3'}$.

The numerator function is found by noticing that

$$E_i = \frac{\Delta_{ik}}{\Delta} I_k \qquad (19)$$

In order to get Δ_{ik} from Δ in expanded form, the column k and row i are cancelled from each determinant of a residual network in the expression. Thus, from Equation 18

$$\Delta_{ik} = S^3L_2C_1C_3\Delta_{ikl2'3} + S^2(L_2C_1\Delta_{ikl2'3'} + C_1C_3\Delta_{ikl23} + L_2C_3\Delta_{ikl'2'3'}) + S(C_1\Delta_{ikl23'} + L_2\Delta_{ikl'2'3'} + C_3\Delta_{ikl'23}) + \Delta_{l'23'} \qquad (20)$$

Of course, if row i and column k had been collapsed previously, the value of the residual determinant in each case will be zero. Note that each determinant in Equation 20 is cofacotr of a determinant in Equation 18. If the row or column subscripted in the expansion of the principal determinant corresponds to i or (and) k, then the determinant value is zero. It can be shown that Percival's method can be applied directly to networks described by the cofactors and determinants in Equations 18 and 20. There fore it is possible to calculate the coefficients of S^K in each driving point and transfer function at one time.

REFERENCES

1. W. S. Percival, Solution of Passive Electrical Networks by Means of Mathematical Trees, Journal, Institute of Electrical Engineers, Vol. 100, Part III, 1953, pp. 143-150.

2. R. H. Seacat, A Method of Network Analysis Using Residual Networks, (Dissertation Texas A&M University).

3. W. Feussner, Uber Stromverzweigung in Netzformigen Leitern, Annalen der Physik, Vol. 9, Fourth Series, 1952, pp. 1355-1329.

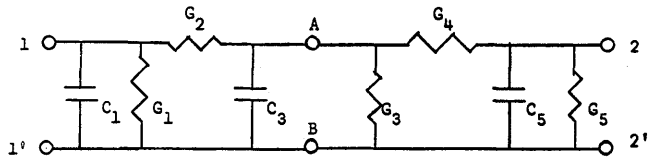


Figure 1. Network Used to Illustrate the Calculation of Δ Using Percival's Method.

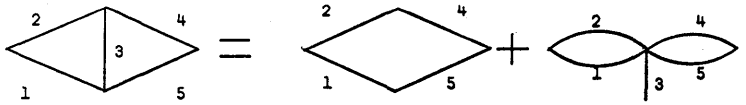


Figure 2. Simplified Graph for Figure 1.

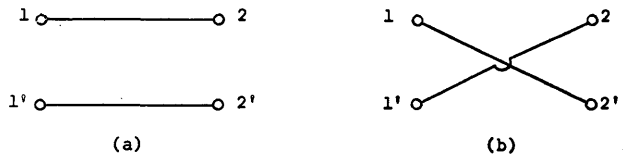


Figure 3. Paths Used to Calculate Δ_{12}.

Figure 4. (a) Graph With Input Shorted;
(b) Graph With Output Shorted.

SUNDSTRAND COMPUTER PROGRAM DESCRIPTION

9-21-64

PROGRAM TITLE: FORTRAN Flow Chart Program

PROGRAM NO.: H-11

PROGRAMMER: R. E. Babcock **ANALYST:** R. E. Babcock

PURPOSE OF PROGRAM:

To prepare by means of 1620 computer a flow chart (block diagram) of any 1620 FORTRAN Computer Program.

INPUT REQUIRED:

Any 1620 FORTRAN source deck up through level FORTRAN II-D. Other non-IBM 1620 FORTRAN languages are also acceptable.

OUTPUT RECEIVED:

Complete flow chart of the program.

REFERENCES:

None.

DISCUSSION:

The FORTRAN Flow Chart Program is designed to accept FORTRAN source decks as input and from this automatically prepare a flow chart or block diagram of the program.

Through the use of the FORTRAN Flow Chart Program a FORTRAN program can be presented in a fashion which is considerably more "readable" from the standpoint of following program logic. This is not only helpful to the original programmer but also anyone put into a position to decipher another programmers work.

The FORTRAN Flow Chart Program may be used in several ways. The primary use would be as an aid in the preparation of intermediate or final program

SUNDSTRAND COMPUTER PROGRAM DESCRIPTION

9-21-64

PROGRAM TITLE (CONT) FORTRAN Flow Chart Program (H-11)

DISCUSSION (CONT.)

documentation. The program also presents a convenient way of keeping documentation current due to modifications or additions which may be required to a program at a later date. In addition, it is also helpful to the programmer during the debugging phase of a program since the output has a way of pinpointing certain logical programming errors as well as keeping an up-to-date flow chart always available.

FEATURES, COMMENTS, AND LIMITATIONS OF THE FORTRAN FLOW CHART PROGRAM

1. Adjustment for number of lines to appear on any one page can be made by changing the variable LENGT in the FORTRAN source deck. LENGT is built in as 74 for a page length of 11 inches and printing at 8 lines/inch. Proportional adjustments should be made from this number in the event a different length form is used or printing is done at 6 lines/inch. Printing at 8 lines/inch should be used where possible as it gives a more compact flow diagram as well as one more pleasing to the eye.

2. An * in Column 7 of a "COMMENT" card will result in the program skipping to the next page to resume printing of the flow chart. This is so that various logical segments of the program can be separated and referred to or used separately if desired.

3. Source decks may be stacked for continual processing as long as each program has as its last card an "END" statement.
Control cards (if any) should be removed from decks for this application.

4. If the first card of the source deck is a "COMMENT" (normally card with program title) it will be repeated as the heading of each new page along with the page number. If the first card of the source deck is not a comment, only the page number will appear at the beginning of each page.

5. The program will handle all 1620 FORTRAN statements through FORTRAN II-D. (Magnetic tape statements excluded). Any unrecognizable statement will be printed as card image (unless it contains record (≠) or group (≢) marks.

6. Statements within a DO or nested DO's will be indicated with + signs to left of statement (see sample flow chart) to indicate that they are within a loop. Up to 19 nested DO's are allowed.

7. Input/output statements will be denoted with 3 leading * as their distinguishing identification rather than enclosure in any sort of box. These statements include READ, RECORD, PUNCH, PRINT, FETCH, and WRITE. TYPE statement will not have leading *'s.

SUNDSTRAND COMPUTER PROGRAM DESCRIPTION

9-21-64

PROGRAM TITLE (CONT) FORTRAN Flow Chart Program (H-11)

DISCUSSION (CONT.)

8. "IF" statements are printed within a diamond-shaped figure of fixed dimension. If the number of characters (excluding blanks) within the parenthesis of an "IF" statement exceeds 17, the statement will overhang the right side of the figure (no characters will be dropped). If there are continuation cards associated with an IF statement, they will not be included within the diamond, but printed following the figure.

9. "GO TO" statements are indicated merely by a small box enclosing the statement number which indicates where logic is to be transferred.

The FORTRAN Flow Chart Program is written in the FORTRAN II programming language. The program has been submitted to the 1620 Users Group Library and is listed under File No. 1.6.118. Program requests through the library will include two source decks, one for 1443 On-line Printer output and one for card output for Off-line printing. Minimum hardware requirements are those as required by the FORTRAN II system. (40 K, Indirect Addressing, card Read-Punch, and Automatic Divide).

Attached are sample output flow charts which were prepared by the program.

FORM 786-266(1/2)

③ 100

SAMPLE INPUT DATA TO FORTRAN FLOW CHART PROGRAM

```
C     GAS BEARING LOADS   E-2                                          002
C     PROGRAMMED BY B. THOMPSON  APRIL, 1962                           003
      DIMENSION A(2),C(2)                                              004
    1 C(1)=1.                                                          005
      C(2)=-1.                                                         006
      E=0.                                                             007
      G=0.                                                             008
      LINK=0                                                           009
      READ 100,G,B,D,E,F                                               010
      READ 101,GINC,GMAX,EINC,EMAX,PCT,PA                              011
      IF(E)14,14,15                                                    012
   14 E=EINC                                                           013
   15 F1=F-1.                                                          014
      EDIF=E-EMAX                                                      015
    2 E2=E*E                                                           016
      AE=1.-E2                                                         017
      P=G/((AE)**3.)**.5                                               018
      IF(LINK)16,16,3                                                  019
   16 H=P                                                              020
    3 H2=H*H                                                           021
      AH=1.+H2                                                         022
      Q=1.-E2*H2/AH                                                    023
      FH=(Q**F)*P-H                                                    024
      S=-2.*F*E2*H/(AH)**2.                                            025
      T=((AH-E2*H2)/(AH))**F1                                          026
      DH=P*S*T-1.                                                      027
      HDH=FH/DH                                                        028
      TOL=PCT*H                                                        029
      IF(ABS(FH)-TOL)5,5,4                                             030
    4 H=H-HDH                                                          031
      GO TO 3                                                          032
    5 U=AH**.5                                                         033
      DO6I=1,2                                                         034
    6 A(I)=((U+C(I))/(2.*AE))**.5                                      035
      R1=3.1415927*F*H/AH                                              036
      R2=(G/P*(1.-AE**.5))/(E*Q**F)                                    037
      R=R1*R2                                                          038
      BD=B/D                                                           039
      U1=2.*A(2)*BD                                                    040
      U2=2.*A(1)*BD                                                    041
      U3=A(1)-A(2)*H                                                   042
      U4=A(1)*H+A(2)                                                   043
      U5=EXP(U2)                                                       044
      U6=EXP(-U2)                                                      045
      V=BD*(AH)**.5*((U5+U6)*.5+COS(U1))                               046
      X1=U3*SIN(U1)-U4*((U5-U6)*.5)                                    047
      X2=U3*((U5-U6)*.5)+U4*SIN(U1)                                    048
      W1=R*(H/AE+X1/V)                                                 049
      W2=R*(1./AE-X2/V)                                                050
      XL=(W1*W1+W2*W2)**.5*PA*B*D                                      051
      IF(LINK)7,7,8                                                    052
    7 LINK=1                                                           053
      PUNCH 107                                                        054
      PUNCH 100,G,B,D,E,F                                              055
      PUNCH 101,GINC,GMAX,EINC,EMAX,PCT,PA                             056
      PUNCH 107                                                        057
      PUNCH 102                                                        058
      PUNCH 103,G,E,W1,W2,XL,H                                         059
```

④ 101

SAMPLE INPUT DATA (CONT.)

```
      GO TO 9
    8 PUNCH 103,G,E,W1,W2,XL,H
    9 IF(EDIF)12,10,12
   10 IF(G-GMAX)11,1,1
   11 G=G+GINC
      GO TO 2
   12 IF(E-EMAX)13,1,1
   13 E=E+EINC
      IF(E-1.)2,1,1
  100 FORMAT(F14.8,F14.8,F14.8,F14.8,F14.8)
  101 FORMAT(F10.4,F14.8,F10.4,F14.8,E7.1,F10.5)
  102 FORMAT(6X1HG,9X1HE,11X2HW1,11X2HW2,10X1HL,10X1HH)
  103 FORMAT(F11.6,F11.6,F13.8,F13.8,F11.6,F11.6)
  107 FORMAT(1X)
      IF(SENSE SWITCH 4)1,2
      END
```



```
                    I
     GAS BEARING LOADS   E-2

     PROGRAMMED BY B. THOMPSON  APRIL, 1962

        DIMENSION A(2),C(2)
                    I
        ---------------------------------------
  1     I C(1) =1.                            I
        I C(2) =-1.                           I
        I E    =0.                            I
        I G    =0.                            I
        I LINK =0                             I
        ---------------------------------------
                    I
     ***READ 100,G,B,D,E,F
     ***READ 101,GINC,GMAX,EINC,EMAX,PCT,PA
                    I
                    IF
                   (E)
     -I             0I              +I
   I 14 I         I 14 I          I 15 I
                    I
        ---------------------------------------
 14     I E    =EINC                          I
        ---------------------------------------
                    I
        ---------------------------------------
 15     I F1   =F-1.                          I
        I EDIF =E-EMAX                        I
        ---------------------------------------
                    I
        ---------------------------------------
  2     I E2   =E*E                           I
        I AE   =1.-E2                         I
        I P    =G/((AE)**3.)**.5              I
        ---------------------------------------
                    I
                    IF
                  (LINK)
     -I             0I              +I
   I 16 I         I 16 I          I 3 I
                    I
        ---------------------------------------
 16     I H    =P                             I
        ---------------------------------------
                    I
        ---------------------------------------
  3     I H2   =H*H                           I
        I AH   =1.+H2                         I
        I Q    =1.-E2*H2/AH                   I
        ---------------------------------------
                    I
```

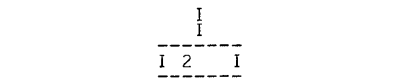
(CONTINUED ON PAGE 2)

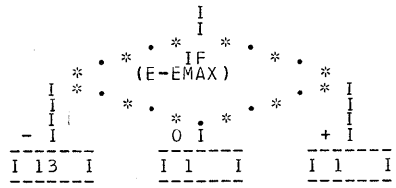
```
                    I
                    I
          ------------------------------------------
          I FH    =(Q**F)*P-H                      I
          I S     =-2.*F*E2*H/(AH)**2.             I
          I T     =((AH-E2*H2)/(AH))**F1           I
          I DH    =P*S*T-1.                        I
          I HDH   =FH/DH                           I
          I TOL   =PCT*H                           I
          ------------------------------------------
                    I
                    I
                    I
                 . * * .
              . *  IF   * .
            *   (ABS(FH)-TOL)  *
          I * .               . * I
          I     * .         . *   I
          I       * . * . *       I
        - I         0 I           + I
        -------     -------       -------
        I 5   I     I 5   I       I 4   I
        -------     -------       -------

                    I
          ------------------------------------------
4         I H     =H-HDH                           I
          ------------------------------------------
                    I
                    I
                    I
                  -------
                  I 3   I
                  -------


                    I
          ------------------------------------------
5         I U     =AH**.5                          I
          ------------------------------------------
                    I
                    I
                 +-------------+
                 + DO          +
      ++++++++++   6           +
      +          + I=1,2       +
      +           +-------------+
      +                 I
      +                 I
      +  ------------------------------------------
6     ++++++++I A(I)  =((U+C(I))/(2.*AE))**.5      I
              I R1    =3.1415927*F*H/AH            I
              I R2    =(G/P*(1.-AE**.5))/(E*Q**F)  I
              I R     =R1*R2                       I
              I BD    =B/D                         I
              I U1    =2.*A(2)*BD                  I
              I U2    =2.*A(1)*BD                  I
              I U3    =A(1)-A(2)*H                 I
              I U4    =A(1)*H+A(2)                 I
              I U5    =EXP(U2)                     I
              I U6    =EXP(-U2)                    I
              I V     =BD*(AH)**.5*((U5+U6)*.5+COS(U1))I
              I X1    =U3*SIN(U1)-U4*((U5-U6)*.5)  I
              I X2    =U3*((U5-U6)*.5)+U4*SIN(U1)  I
              I W1    =R*(H/AE+X1/V)               I
              I W2    =R*(1./AE-X2/V)              I
              I XL    =(W1*W1+W2*W2)**.5*PA*B*D     I
              ------------------------------------------
                        I
                        I
```

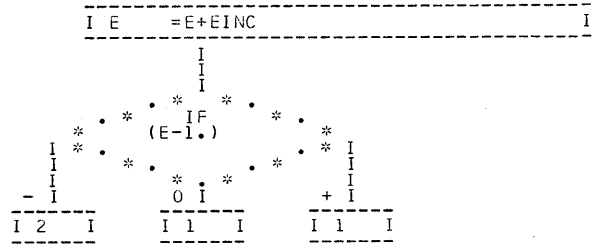
(CONTINUED ON PAGE 3)


```
                    I
                    I
                    I
                 . * * .
              . *  IF   * .
            *   (LINK)       *
          I * .               . * I
          I     * .         . *   I
          I       * . * . *       I
        - I         0 I           + I
        -------     -------       -------
        I 7   I     I 7   I       I 8   I
        -------     -------       -------

                    I
          ------------------------------------------
7         I LINK =1                                I
          ------------------------------------------
                    I
          ***PUNCH 107
          ***PUNCH 100,G,B,D,E,F
          ***PUNCH 101,GINC,GMAX,EINC,EMAX,PCT,PA
          ***PUNCH 107
          ***PUNCH 102
          ***PUNCH 103,G,E,W1,W2,XL,H
                    I
                  -------
                  I 9   I
                  -------


8         ***PUNCH 103,G,E,W1,W2,XL,H
                    I
                    I
9                . * * .
              . *  IF   * .
            *   (EDIF)       *
          I * .               . * I
          I     * .         . *   I
          I       * . * . *       I
        - I         0 I           + I
        -------     -------       -------
        I 12  I     I 10  I       I 12  I
        -------     -------       -------

                    I
                    I
10               . * * .
              . *  IF   * .
            *   (G-GMAX)     *
          I * .               . * I
          I     * .         . *   I
          I       * . * . *       I
        - I         0 I           + I
        -------     -------       -------
        I 11  I     I 1   I       I 1   I
        -------     -------       -------

                    I
          ------------------------------------------
11        I G     =G+GINC                          I
          ------------------------------------------
                    I
                    I
```

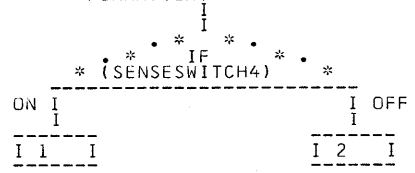
(CONTINUED ON PAGE 4)








```
100         FORMAT(F14.8,F14.8,F14.8,F14.8,F14.8)
101         FORMAT(F10.4,F14.8,F10.4,F14.8,E7.1,F10.5)
102         FORMAT(6X1HG,9X1HE,11X2HW1,11X2HW2,10X1HL,10X1HH)
103         FORMAT(F11.6,F11.6,F13.8,F13.8,F11.6,F11.6)
107         FORMAT(1X)
```



END

SUNDYNE PUMP

QUOTING SYSTEM

TABLE OF CONTENTS

PREPARED BY:

Robert E. Babcock
R. E. Babcock

F. Bollenbach
F. Bollenbach

SUNDSTRAND AVIATION - DENVER
Denver, Colorado

I. GENERAL DESCRIPTION OF SUNDYNE PUMP QUOTING SYSTEM

A. PURPOSE AND GENERAL DISCUSSION

The Sundyne Pump Quoting System is a fully computerized system which has the purpose of accepting input from potential pump buyers and designing and selecting pump components, and (1) printing all of the pump components, materials of construction, operating parameters, etc., on a pump specification sheet; (2) giving a performance graph of the pump by means of Plotter output; and (3) printing a complete pricing sheet. The three items above then constitute a quote which can be sent directly to the potential customer via the sales representatives. The system was designed in coordination with the Industrial Products Sales Department, Engineering Department, and the Computer Department.

The Sundyne pump is an open impeller, single stage, centrifugal pump. Diffusion is done in a single conical diffuser rather than in a collecting scroll which is typical of conventional centrifugal pumps. This design offers better efficiency in the low flow, high head region. Gearboxes of various speeds are added when required for the higher head applications. The major components designed and/or selected by the program include impeller, diffuser, inducer, gearbox (speed), and motor.

The Quoting System was designed and implemented in order to alleviate the application engineer from tedious manual pump designs which were required to handle the many requests for pump quotes coming in from the field. The system gives a complete, accurate, consistent, and optimum pump selection resulting in Sundstrand being able to quote the most competitive pump possible. The system is also used by engineering to handle pump orders as well as to experiment with more optimum pump configurations.

The Quoting System makes use of the IBM 1620, Model II Computer, 1311 Disk Files, On-line 1443 Printer, and an IBM 870 System Plotter.

All basic pump design parameters, materials of construction, prices, and all reference design information is stored on disk packs and is available to the various components of the program. This information can be updated at any time were there changes in pump design constants, vendor information, prices, item descriptions, etc.

The programming system is separated into various individual programs, all of which are tied together and 'called' independently, but automatically under program control. The components of the program consist of:

1. Pump Selector Program
2. Pump Performance Plot Program
3. Pump Specification Program, and
4. Pump Pricing Program.

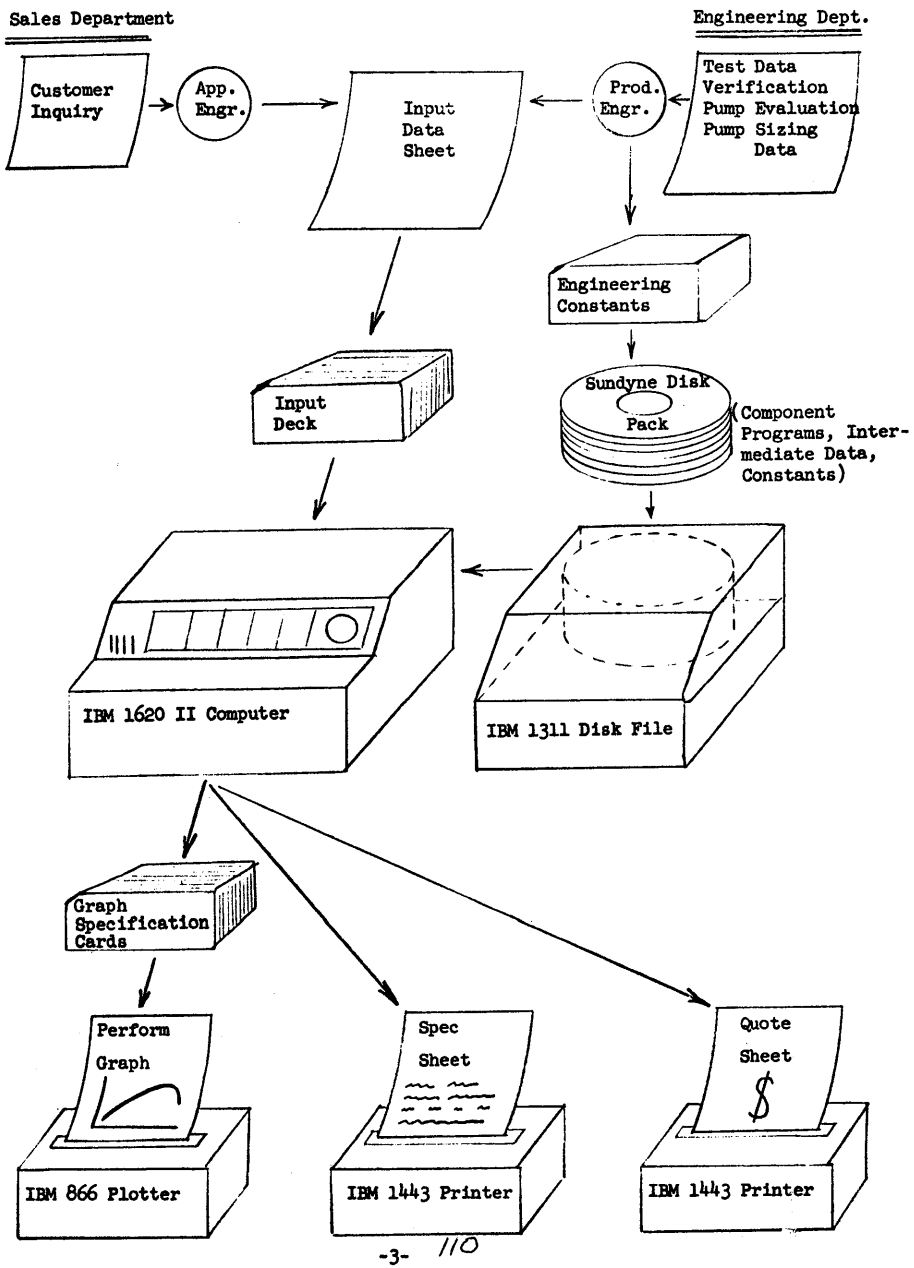
Each of these programs is also stored on disk and 'called' when required. A fifth program is also available for the purpose of updating information in the disk packs, but is only used when this requirement is needed.

The system has the ability to process up to 200 quotes at a time. Each quote is processed in approximately one minute on the 1620 Model II computer. The equivalent time by hand was approximately two hours. The disk packs are also used to accumulate intermediate data between the program components. This speeds up the operation of the system since each program component may be called in only once for each running of a group of quotes.

B. SYSTEM SCHEMATIC

Following is a schematic which depicts the overall flow of information from inquiry to quote documents. This schematic portrays the overall tie-in concerning the use of the program and the computing equipment required to prepare the quote.

SYSTEM SCHEMATIC



C. SAMPLE DOCUMENTS

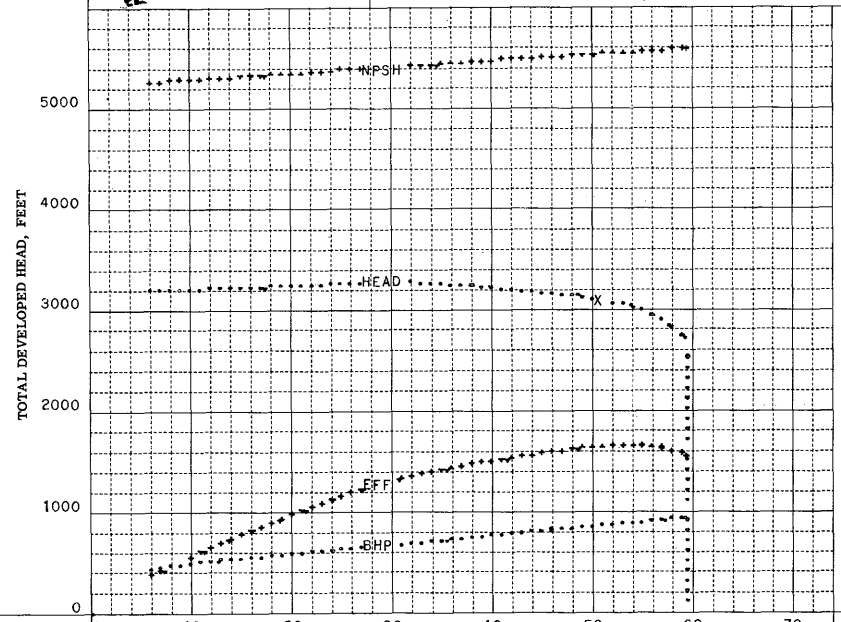
As far as the computer program is concerned a pump quote consists of the following documents:

1. Performance Graph
2. Specification Sheet
3. Pricing Sheet.

Samples of these output documents follow. As is noted, each document is printed on a special pre-printed form. A sample of the input data sheet from which these sample output sheets were generated is also included. Depending upon the input data received, the program may generate up to three separate quotes for each set of input data. The application engineer must select the most desirable one as the actual quote to be sent to the customer.

CUSTOMER X·Y·Z· MANUFACTURING CO·

PUMP MODEL LMV-3 RPM 24727· DATE 7 31 64 BY



SUNDSTRAND

SUNDYNE PROCESS PUMPS

CURVES ARE BASED ON SHOP TESTS HANDLING COOL WATER AT A SPECIFIC GRAVITY OF 1.0

ITEM NO. 16·

SERIAL NO.

QUOTE NO. 1788

11.2

SUNDSTRAND

SUNDYNE PUMP SPECIFICATION SHEET

Job No. 5327·
Item No. 16·
Spec. No. 3H46
Quote No. 1788
P. O. No. 433P
Date 7 31 64
By

GENERAL

FOR X.Y.Z. MANUFACTURING CO. NO. REQ'D 5
LOCATION SITE 433, PEORIA
SERVICE CHEMICAL FLUSHING

OPERATING CONDITIONS

Liquid Pumped WATER @ 120. °F P. T.
Capacity: GPM design 50 ; GPM Normal 50.
Specific Gravity .930 ; Viscosity 1. cps; Vapor Pressure 9. psia
Corrosive Material? Errosive Material?
Disch. Press. 1248 psig; Suct. Press. 0 psig; Diff. Press. 1248 psi; TDH 3100 ft.
NPSH: Avail 14.0 ft. (liquid); Req'd ft. (liquid); Req'd 13.6 ft. (water)

PUMP SPECS.

Size & Type 3X2 LINE MTD. VERT. Model LMV- 3 ; Bhp: Design 88.0 Max. 96.6
Suction: Size 3 ; ASA rating 300 ; Facing RF ; Driver 100.0 hp; Design Efficiency 41.3%
Discharge: Size 2 ; ASA rating 600 ; Facing RF ; Outline Drawing No. ENCLOSED
Impeller Dia: Design 3.47 Max 5.06 Min 3.00 Sectional Drawing No. ENCLOSED

MATERIALS OF CONSTRUCTION

Case 3.5 NI
Shaft 4615
Inducer 316SS
Seal Rotating Face TUNGSTEN CARBIDE
Seal Gaskets TEFLON
Diffuser CARBON STEEL
Shaft Sleeve 316SS
Throttle Bushing GRAPHITAR
Seal Stationary Face GRAPHITAR 39
Pump Gaskets TEFLON
Impeller 316SS
Seal Housing 3.5NI
Seal Manufacturer SEALOL
Other Seal Parts 300 SERIES SS

TESTS REQ'D

YES NO
☐ ☐ 1. NPSH
☐ Other (Describe)
☒ ☐ 2. Performance

(NPSH &/or witnessed testing available at extra cost upon Customer Request.)

GEAR BOX LMV-2 ONLY

Input RPM 3550 Output RPM 24727. Lubrication: Lt. Turbine Oil
Oil Heat Exchanger Required: Yes. X No

ELECTRIC MOTOR

Manufacturer GENERAL ELECTRIC ; H.P. 100.0 ; Frame 405 UP Phase 3
Cycle 60 ; Volts 440 ; Enclosure EXPLOSION PROOF
RPM 3550 ; Bearings BALL ; Lubrication GREASE

STEAM TURBINE

Manufacturer ; Type ; HP
Steam Pressure: Inlet psig; Exhaust psig ; RPM
Steam Quality Water Rate Lbs. per HP-hr.

WEIGHT

NET WEIGHT: PUMP 320 Lbs. · GEARBOX 210 Lbs. · MOTOR 1575 Lbs.

SHIPPING WEIGHT: Pump and Driver 2185 Lbs.

REMARKS

2 TO 3 GPM OF 60-120 DEG F FLUID MUST BE CIRCULATED THROUGH THE GEARBOX OIL HEAT EXCHANGER

113

QUOTATION

SUNDSTRAND - DENVER

DIVISION OF SUNDSTRAND CORPORATION
INDUSTRIAL PRODUCTS GROUP

2480 WEST 70th AVE. • DENVER, COLORADO 80221
TELEPHONE 428-3636 AREA CODE 303 TWX 303-428-3636

TO: X.Y.Z. MANUFACTURING CO.
2100 S. MICHIGAN AVE.
CHICAGO, ILLINOIS

QUOTATION NO. 1788
PLEASE REFER TO THIS NUMBER ON YOUR PURCHASE ORDER.

DATE 7 31 64

ATTENTION: MR. JOHN SMITH

YOUR INQUIRY NUMBER 64A-1093

ITEM	QUANTITY	DESCRIPTION	
16.	5	LMV-3 SUNDYNE PROCESS PUMP COMPLETE WITH 100.0 HP MOTOR, GENERAL ELECTRIC EXPLOSION PROOF PUMP CONSTRUCTION CARBON STEEL 24727. RPM GEARBOX INDUCER LUBE OIL HEAT EXCHANGER PRICE EACH - $ 6709.00 FREIGHT EACH - $ 24.03	$ 33545.00 $ 120.17

DELIVERY _____ WEEKS AFTER RECEIPT OF ORDER.

TERMS: NET 30 DAYS.

F.O.B. POINT DENVER, COLORADO WITH FREIGHT ALLOWED TO SITE 433, PEORIA

THIS QUOTATION SUBJECT TO THE TERMS AND CONDITIONS OF SALE APPEARING ON THE REVERSE SIDE HEREOF.

ACCEPTANCE

FOR _____ DATE _____

BY _____ TITLE _____

SUNDSTRAND-DENVER

BY _____ TITLE _____

PRICE BREAKDOWN

PUMP	700.00
MOTOR	2094.00
GASKET	25.00
METALLURGY	250.00
SEAL	50.00
GEARBOX	3240.00
INDUCER	250.00
HEAT EXCHANGER	100.00
TOTAL	$ 6709.00

114

E. GENERAL DESCRIPTION OF SYSTEM COMPONENT PROGRAMS

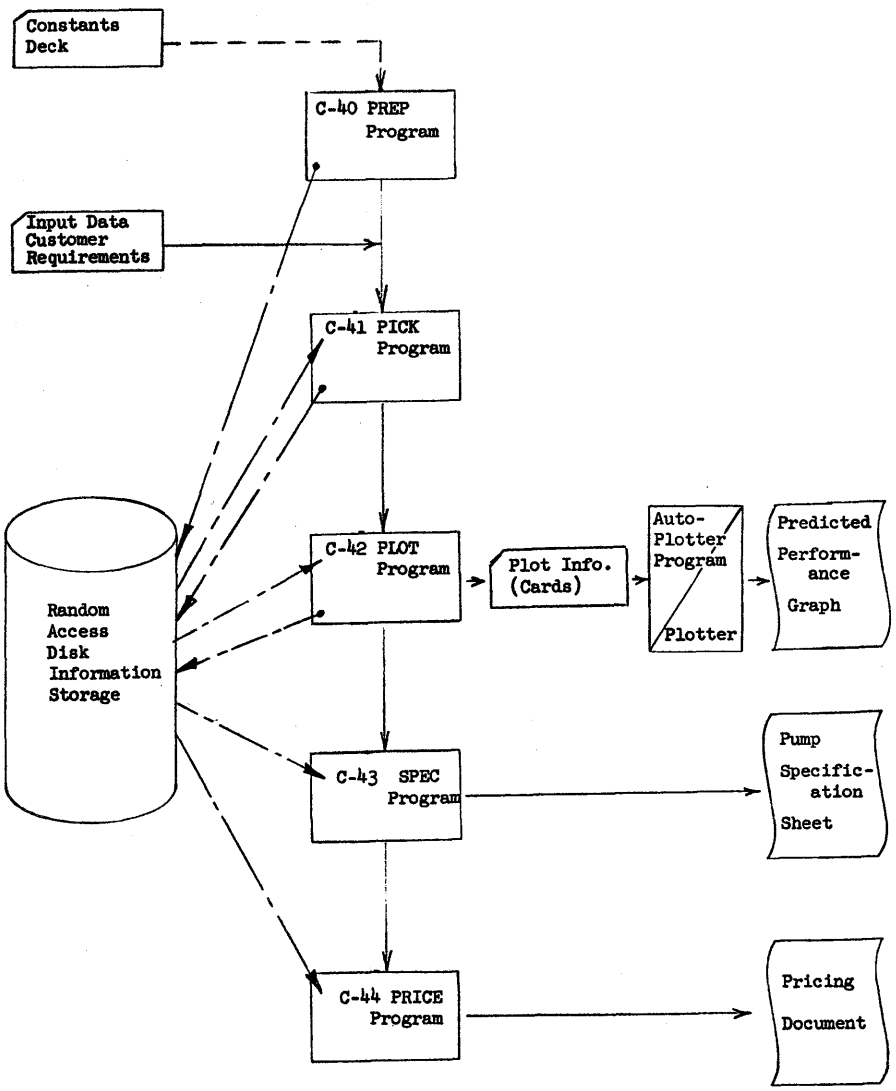
The Sundyne Pump Quoting System consists of five (5) separate component programs. Four of these programs - Pick (C-41); Plot (C-42); Spec (C-43); and Price (C-44) are used every time a quote is made. These programs are each stored permanently in disk storage and are called automatically into computer memory for execution in chain fashion as required in the above order.

When more than one quote is processed (program may process up to 200 quotes at a time), the intermediate data required between component programs for each quote is stored on disk. In this way each component program need be called into memory only once for each group of quotes run. This speeds up the overall execution time and thus is the reason for the maximum allowable of 200 quotes at one time (intermediate disk storage limitation).

The fifth component program (C-40) is used only to load or restore the data package into disk storage.

The following macro chart illustrates the general scheme of how each program is run through the computer. A brief explanation of each component program follows.

SYSTEM MACRO CHART



SYSTEM EXPLANATION

CONSTANTS DECK:

Numeric constants are the most current engineering equations, limits, graphs, tables, etc., of pump performance. Alphabetic phrases are the most current information on materials of construction of gaskets, seals, pumps, motors and the pricing of these components.

INPUT DECK:

Customer's numeric data of operating conditions. Customer's alphabetic data of name, address, type of liquid, and service, additional numeric coding of materials of construction and identification numbers.

C-40:

This program reads the constants deck and stores the constants in specific sectors on the disk for use in C-41, C-42, C-43 and C-44. It is used only when new constants are to be stored and not needed with each data run.

C-41:

This program reads each customers requirements from the input deck and designs a pump or pumps with various combinations of inducers, gear boxes, motors, as necessary to satisfy these requirements. The input data and design point data are stored on the disk and a counter maintained of the number of customers completed.

C-42:

This program fetches the input data and design point data for each pump from the disk and calculates the performance parameters over the full range of flow and punches these calculations on specification cards for use in the autoplotter program.

C-43:

This program fetches the input data and design point data and the materials of construction codes for each pump from the disk and prints out all parts information on a preprinted form called the SPECIFICATION SHEET.

SYSTEM EXPLANATION (continued)

C-44:

This program fetches the input data and design point data, materials of construction codes, and quantity to quote for each pump from the disk, and calculates unit and total prices and prints this information on a preprinted form called the QUOTE SHEET.

AUTOPLOTTER:

This program reads prepared specification cards and punches plotting commands on new cards for use in the plotter to plot the pump performance on preprinted forms called the PERFORMANCE GRAPH.

LOUISIANA TECH INFORMATION RETRIEVAL SYSTEM

David A. Hake, Director-Computing Center
Louisiana Polytechnic Institute
Ruston, Louisiana

INTRODUCTION

This paper describes an Information Retrieval System in use at Louisiana Tech implemented for an IBM 1620 Model 1 with 40K to 60K memory, indirect addressing, edit commands and one or more 1311 Disk Storage Drive. The system is unusual in (1) the power provided the user through the use of "AND-OR" logic in search request formation, (2) a search phase requiring only the operation of division and (3) the ability to provide for more comprehensive libraries or interdisciplinary searches via interchangeable 1311 disk packs.

The system is discussed in two parts. Part I from the user point of view, describes the composition of the document library, the method of abstracting, disk library concept and system design as preparation for Part II of the paper which delves into the "how" aspects of the system.

FRAME OF REFERENCE - PART I

How many times have you sat pondering a problem or a situation and recalled reading an article last week, or last month...or maybe it was yesterday, whose contents dealt directly with the topic at hand? Thinking further, you recall the illustrations given, the size of the pages, that the author had an unusually long last name, some topical headings and that the third paragraph was rather ambiguous. The name of the publication? The issue? The author's name?..... Obfuscation!

Thinking about thinking of this dilemma, we have been able to rationalize that some day our friends, the machines, will enable us to bypass these periodic frustrations. Eventually, a desk top device will provide ready access to the nearest electronic information store. However, as we are not yet at this stage of development, more mundane approaches prevail.

Computer techniques for Information Retrieval, while not a panacea for the varied types of information problems, are generally accepted as a step in the right direction. It is further generally accepted that the task of automated storage and retrieval of documents is limited to the larger general purpose computers or those machines specifically designed for that purpose. Several factors have fostered the large system bias.

1. Large Library concept or the capacity to search a vast store of documents. Is it valid to assume a system is worthless unless 100,000 or more documents comprise the search set? As a majority of information users do not have access to such a comprehensive store, they are resigned to the manual search entirely. In many instances, a relatively small, specialized document set accessable through widely used hardware would provide a valuable tool with greatly reduced economic requirements for both implementation and practical use of the system.
2. Alphabetic nature of information retrieval source data has produced a host of search techniques all requiring a high degree of non-productive alphabetic manipulation prior to actual comparison. To offset such time consuming manipulation, high internal speeds, as provided in large scale systems, are required to reduce search time to "reasonable" proportions.
3. Sophisticated abstracting schemes -- The use of auto-abstracting techniques in which free text is analyzed via computer to produce an abstract or descriptors, does dictate a large-scale system. Such reduction to eliminate human bias and provide uniformity in analysis of text precipitates yet another problem; that of committing the full text to a machine readable media. Until optical character readers or methods of machine readable tapes produced during typesetting are economically feasible, this technique is available only to the extremely well endowed installation.

The rationale of the small-scale retrieval system is not antithetical with that of the large-scale system. Rather, it is composed of justifiable compromise in the method of abstracting and the size and composition of the document set.

THE DOCUMENT SET

Generally, the pulse of the research community, or current developments within a particular field of endeavor, is best monitored through the industry periodical or journal as opposed to text and other multi-topic hardbound copy. Using the keyword-in-context (KWIC) method for selecting document descriptors, the Tech system is mainly, though not exclusively, intended for retrieval of periodic literature.

Where these periodical publications are germain to a particular field of endeavor, e.g., physics, agricultural engineering, gas utilities, etc., they are treated as an entity by this system. Such a specialized collection of documents is termed a "document-set."

As document sets are manipulated independently, many keywords or descriptors common to each member of the set may be eliminated from the combined "descriptor list." (Herein termed dictionary.) For example, articles of an agricultural engineering document-set need not show agriculture or engineering as keywords. Elimination of like terms speeds abstracting, keypunching, searching, and reduces internal storage requirements per document.

Although it is not the purpose of this paper to investigate the linguistic problems inherent in the use of KWIC abstracting, they are recognized as a major influence upon the ultimate worth of this or any document retrieval system. Sufficient barriers to retrieval exist in the more common problems such as plurals, prefixes, suffixes and synonyms. Basically the problem arises when the abstractor must choose the form of a word to enter as a descriptor. Consider an article Matrix Inversion and Related Topics. From the text and the abstractor's own

experience, a list of keywords is prepared. (Figure 1.) More than a few KWIC systems require that all words of a user's search argument be present in a document's keyword list in order for that document to be retrieved. A dilemma arises immediately as to which form of the words to use on the final keyword list.

One solution would be to include all possible forms and synonyms of each word in the keyword list. At best this is a difficult task. In this example, twenty keywords would be entered to express just ten basic concepts. Even with 20 keywords, a simple search argument composed of METHODS, MATRIX, INVERSION would be unsuccessful due to the failure of the abstractor to include METHODS as another synonym in the keyword list.

"Thesaurus-like" responsibility on the part of the abstractor is reduced in this system by allowing the user flexibility in the statement of the search request. The search is phrased as a group of concepts rather than collection of words. By removing this burden the abstractor can concentrate on more effective keywording, giving less thought to synonyms and word endings.

ABSTRACTING EXPERIENCE

In order to assess the relative effectiveness of our keywording efforts, abstracting forms were sent each author appearing in Volume 10 of the Journal of the ACM with the request that the author keyword his own article. Instructions given the authors were purposefully held to a minimum. Each was told only that the KWIC scheme was to be used and that any number of words up to 20 would be acceptable. Thirty out of forty-two inquiries were returned.

Results of the study are shown in Figure 2. The number of keywords per document used by the authors ranged from 4 to 15 with the mode at 9 and average of 9.4 words. Abstractor selections also ranged from 4-15 with mode at 8 and an average of 8.3 words per document. Significantly, more than half the words used by the abstractor were also used by the author. It is important to note, however,

POSSIBLE KEYWORDS
FROM DOCUMENT ON MATRIX INVERSION
AND RELATED TOPICS

ABSTRACTOR'S DILEMMA

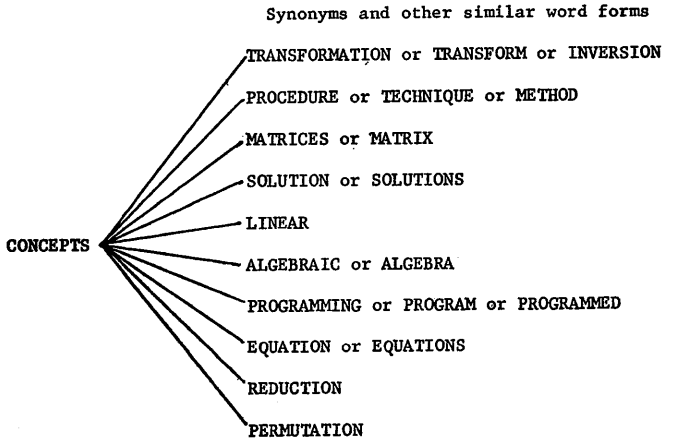


Figure 1

KWIC Comparison
on Documents from Journal of ACM, Vol. 10, 1963

NUMBER OF KEYWORDS PER DOCUMENT

	AUTHOR SELECTED	ABSTRACTOR SELECTED	AGREEMENT ON
RANGE	4 to 15	4 to 15	2 to 7
MODE	9	8	5
AVERAGE	9.4	8.3	4.4

Figure 2

that of the non-matching words, a majority were of a synonymous nature. Further, the matching words were found near the first of each list. Since there is a natural tendency to list the more pertinent concept descriptors first, the abstractor's "score" (as to categorization of the article and its subsequent successful retrieval) is appreciably greater than the 50 percent as indicated by number count.

As a by-product of this survey, the authors' and original abstractor's keywords have been edited into a more nearly representative list of keywords for the Volume 10 articles. This will be used to train additional abstractors and serve as a key by which abstracting skill may be assessed.

LIBRARY CONCEPT

As previously stated, a group of topically homogeneous documents comprises a document set, or more simply, a "library." Multiple libraries may be maintained on individual 1311 Disk Packs, however, only one library may be searched at a time. Since the search program resides in core, disk packs (or libraries) may be changed at will and the same search argument applied to several libraries or the extension of a single library onto two or more packs. Minor changes in coding will allow for a multiple drive system which would result in decreased physical handling of packs required for extended libraries or interdisciplinary searches with the same argument.

SYSTEM DESIGN

Four functional, interrelated sections make up the Tech IR system.

UPDATE -- storage

TRANSLATE -- decode search request

DIFFERENTIATE -- search and retrieval

UTILITY PACKAGE

Each will be considered in turn from a user's standpoint.

Update Functions include: disk initialization, storage of initial library and addition of new documents to the system without reloading the original set.

Input at this stage is the bibliographic data, keyword descriptors and document number as shown in Figure 3. The "librarian" assigned document number is selected from a table of primes provided by a Utility program. All documents of a particular library are identified by a unique prime number which serves to relate the bibliography with its keywords. The order of punching is: document number and bibliographic data (not exceeding 100 characters) on one or two cards; up to 32 keywords entered four per card with a period in column 80 of the last card.

After all documents have been loaded, console typewriter output provides information as to disk storage allocation. At this point, the system is ready for use by the public.

Translate This portion accepts the search request as formed by the user.

To aid in composing the request, Utility Program Dictionary provides the user with a list of all keywords currently in the system. Normally, the Dictionary program is run after each Update in order to reflect new words found in the set just added to the library. Figure 4 shows a typical search argument in concept form.

A search is desired on techniques of matrix inversion. Three sets of documents are pertinent: (1) those on matrices, (2) inversion and (3) techniques. Through use of the Dictionary provided the user, he finds other words which might also classify the information and prepares the search accordingly.

Concept A -- MATRIX or MATRICES
AND
Concept B -- INVERSION or TRANSFORM or TRANSFORMATION
AND
Concept C -- TECHNIQUE or METHOD or PROCEDURE

00229
DOCUMENT PRIME

Abstracted by _____

PLEASE PRINT ALL INFORMATION:

BIBLIOGRAPHIC: Be as complete as possible. Standard abbreviations are desirable.

TITLE ALGORITHM - MIN. COST PROCEDURES FOR BINARY DECISIONS

PUBLICATION JOURNAL OF THE ACM

AUTHOR SLAGLE, J.R. PUBLISHER _____

Vol: 11 No: 3 Month: JUL Year: 64 Page: 253

KEYWORDS

Print in any order, as many descriptors as are necessary. Star (*) those descriptors which could possible be omitted. The first keyword must be the author's last name.

1. ~~(author)~~ ALGORITHM
2. MINIMUM
3. COST
4. PROCEDURE
5. BINARY
6. DECISION
7. BOOLEAN
8. PROBABILITY
9. THEOREM
10. _____
11. THEOREM
20. _____

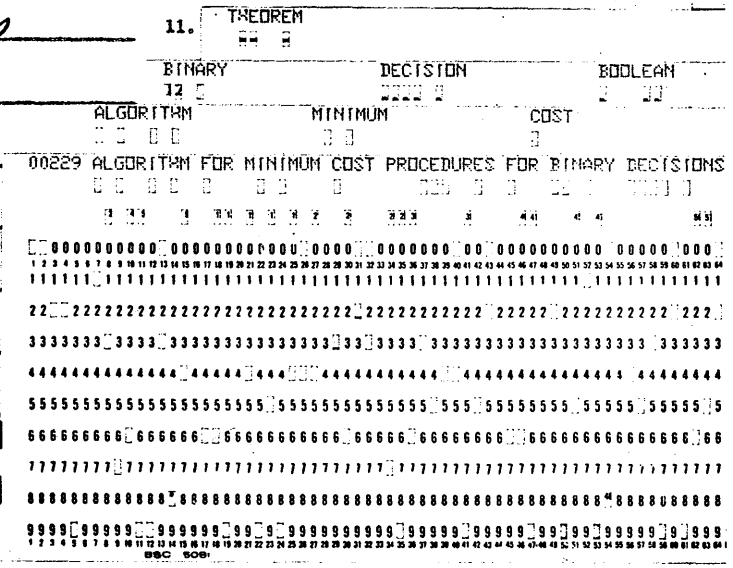


Figure 3

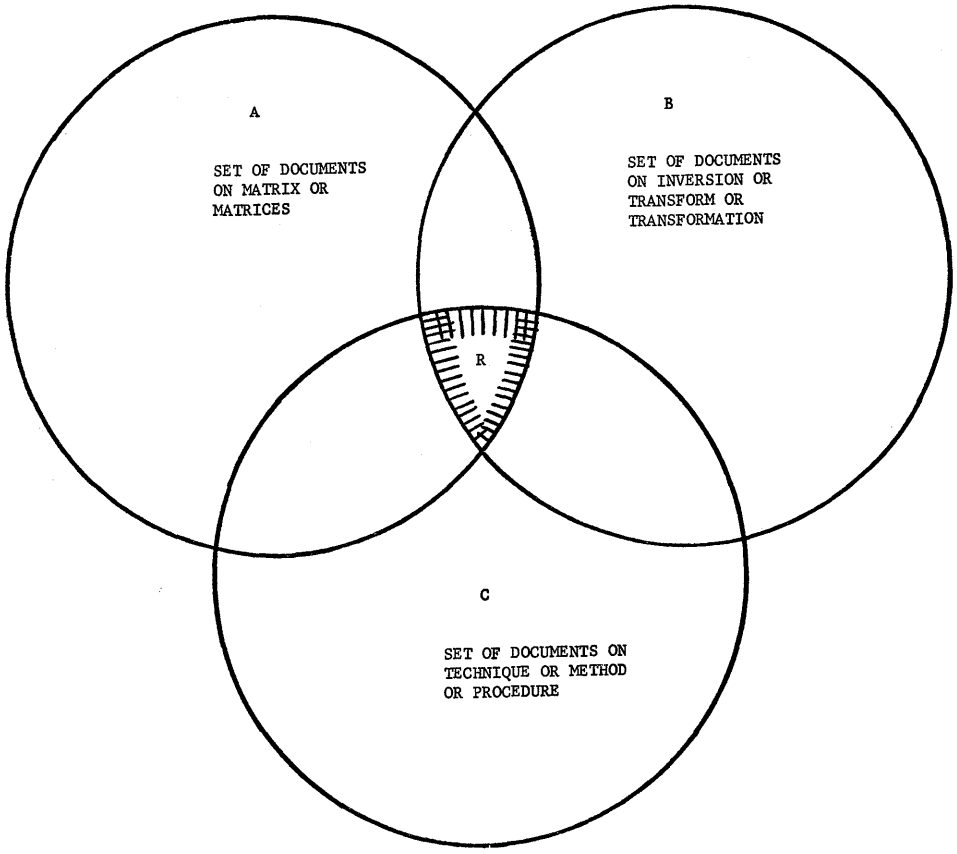


Figure 4
I-10

Documents retrieved at R will contain at least one word from each of concepts A, B, and C.

Through this technique, the user is not left entirely at the mercy of the thought processes of the abstractor who may or may not have included ample synonyms and word permutations, nor does the user require the counsel of those abstractors for formation of his search request.

Another advantage of the request technique is manifest in the possible combination of two or more requests into one search. With reference to the foregoing example, User #2, requiring documents on Laplace transforms could share the search with User #1 by appending "or LAPLACE" to concept A. Requests thus combined require little additional search time.

The system is designed to accept one, two or three words per concept with a maximum of ten concepts. Input is one concept per card with a maximum of three words per card. Words as expressed on a card are separated by implied OR logic while the several cards are separated by implied AND logic.

Output of the Translate phase is the search argument in a numeric form.

Differentiate Accepting the numeric search argument, this program will determine which documents are to be retrieved. At this point the user has the option to receive as output either the document number at the typewriter or the corresponding bibliographic information punched into cards. The document number output option allows the user to note the quantity of documents retrieved. If this quantity is thought excessive, the search may be rerun adding definitive concepts (AND situations) or by removing several of the OR situations from existing concepts. On the other hand, a limited amount of output may prompt the user to delete one or more concepts, or by consulting the Dictionary, supply the maximum number of synonyms and different word forms as OR additions to existing concepts.

Utility Routines Supplementary programs used to date include:

1. Dictionary preparation program punching all keywords entered into the system one per card suitable for sorting and listing.
2. Prime number generator with output in table form from which the abstractor assigns document numbers.
3. Pre Update checking program insures proper and unique assignment of document numbers. When keywords are punched directly from published descriptor lists, the routine checks for erroneous inclusion of duplicate descriptors.
4. File inversion program, not yet completed, will use the title as included in the bibliographic data to punch an inverted file listing.

1620 FORTRAN IV

A Fortran IV System for the IBM 1620 Computer with Magnetic Tapes

by

James White*

Mayo Computer Facility

Mayo Foundation

Rochester, Minnesota

This investigation was supported in part by a U.S.P.H.S. research grant, No. FR 00007, fron the division of research facilities and resorces, N.I.H.

Special thanks is given for the assistance of other staff members of the Mayo Foundation Computer Facility.

* Present address: Marquette University, Milwaukee, Wisconsin.

Virtually all fields of current scientific research, as well as most other numeric operations, are dependent on digital computers. Yet, as recently as ten years ago, computers played a negligable role in research and in most other areas.

One of the most significant factors contributing to the rapid spread of the use of computers has been the development of problem oriented programming languages, designed to reduce the human time and knowledge necessary to write computer programs. The most widely used of these languages is Fortran.

From the first limited versions introduced about ten years ago, Fortran has evolved into a language useful for an increasingly great variety of problems. The most recent IBM version, Fortran IV, includes most features of earlier Fortran versions. In addition, Fortran IV incorporates features of other programming languages, such as Algol. The resulting amalgamation is a language which is better for most purposes and, at the same time, causes minimal obsolescence of previous Fortran programs.

Fortran IV is the computational programming language implemented on nearly all current computers, including the 7040, 7090, 1401 and 360. The lack of a Fortran IV system for the 1620 has been the only significant exception in IBM's development of this "universal" language.

About a year ago, specific planning began to replace the 1620 used in the research computer facility at Mayo Foundation with a 7040. In order to educate open shop users, so that the 7040 could be used as soon as delivered, the decision was made to produce a Fortran IV system for the 1620. This system would have the additional advantage of minimizing the re-programming effort required to run 1620 programs on the 7040.

Many of the specifications for the Fortran IV system were determined by a committee of open shop programmers. This group effort produced a system suitable for use in a variety of areas. Discussion by the programmers of various system philosophies resulted in a group which was well educated in the functions of the new system and, quite significantly, more understanding of the problems of the

systems programmer. The programming of the Fortran IV system was done by persons who were to be closely connected with the operation of the 7040; this programming gave them detailed knowledge of several areas of the new system.

The resulting Fortran IV system has been used, with occasional modifications, since May, 1964. Because of its greater flexibility and more rapid operation, it has been used for almost all new programs written in recent months. Thus, it has proven successful not only as an educational tool, but also as an operating system.

The 1620 Fortran IV system as presently available requires a 60,000 digit memory, 3 magnetic tape drives, floating point hardware, and all Fortran II hardware features. Therefore, the system as it exists now would not be useful for most 1620 installations. This presentation is not a system explanation for probable future users. However, this programming system is presented as an introduction to Fortran IV, and as a source of ideas for anyone contemplating modification of a present compiler system.

For those interested in using Fortran IV, the present system could be modified with moderate effort to require only those hardware features required by Fortran II, or to operate on a 20K disk system. System information has been sent to over thirty 1620 users, some of whom are planning modifications to adapt the system to their own configurations. The session this afternoon will cover the programming system in detail and provide answers to some questions concerning possible system modifications.

The present 1620 Fortran IV compiler is magnetic tape oriented. To initiate compilation a control card is loaded at the 1622. This results in the loading of Pass I from a library tape; this input requires about 6 seconds. Source statements punched on cards are then read, checked for errors, and coded into a string language (intermediate output) which is written onto a second tape.

Provided no errors are found, the Pass II compiler is automatically loaded from the library tape. This compiler then reads the string language program and writes the object program on the third tape, or on cards, or both. (The output medium for the object program is specified by the initial control card). At the conclusion of Pass II, the Pass I compiler is automatically loaded from the library tape. Further programs and subprograms may then be compiled. If the object programs are being written on the third (object program) tape, it is permissible to use previously compiled object decks as input in place of source programs.

At the conclusion of compilation, another control card causes the loader to be entered from the library tape. The loader reads the tape containing the object programs and, after loading these programs, automatically loads the Deck III subroutines from the library tape. It is also possible to load previously compiled object programs by entering control cards which will switch from reading tape to cards or from one tape to another.

The 1620 Fortran IV programming language is essentially the same as Fortran II, except for the changes which will be explained later. 1620 Fortran IV includes all major programming features of 7040 and 360 Fortran IV. With a few restrictions, such as alphameric representation and library subroutines, many types of programs can be run on either system. The language and other programming features which will be discussed are listed on the sheets distributed with this talk.

Let us now review the Fortran IV features included in the 1620 version presently in use. Mode specification statements are new features of Fortran IV which greatly extend its flexibility. These introduce the following changes in terminology. Variables called "fixed-point" in Fortran II are referred to as "integer" in Fortran IV. Those called "floating-point" in Fortran II are designated as "real" in Fortran IV. A third mode, "logical", is

also included in the 1620 Fortran IV specifications.

If the mode of a variable, statement function, or subprogram function is not specified, the rules of Fortran II (whether the first letter is between I and N) are used to determine its mode. However, declarative statements beginning with the word INTEGER or REAL may also be used. The mode name is followed by one or more variable names; these variables will assume the mode declared, superceding the name convention. For example, the first statement

```
REAL I, MARS
INTEGER A, X(20,5)
```

would cause I and MARS to be real (floating point) variables. Dimension specifications may also be included in these declarative statements. The second statement defines X to be an integer array containing 20 times 5 positions.

A third type specification statement is LOGICAL. This will cause the variables listed to be considered logical variables, which can have only the values true or false. Fortran IV includes two logical constants; these are written in source language as .TRUE. and .FALSE. The statements

```
LOGICAL B, I
B = .TRUE.
I = .FALSE.
```

declare B and I to be logical variables, give B a value of true, and give I a value of false.

Logical values may also be created by comparisons of numeric quantities. Six relational operators are included in the Fortran IV language. They are

.LE. less than or equal to

.LT. less than

.EQ. equal to

.NE. not equal to

.GT. greater than

.GE. greater than or equal to

Those relational operators may connect any pair of integer constants, variables, or expressions, or any pair of real constants, variables, or expressions. The resulting logical expression has possible values of true or false. Examples are:

```
LOGICAL D,L
D = B.GT.C
L = (B/C)**7.6+F*G.GE.17.5
```

If the algebraic value of B is greater than the algebraic value of C, D is given a value of true. If C is equal to or larger than B, D becomes false. The periods before and after logical operators and constants serve to distinguish them from Fortran variables.

Three boolean operators, namely .AND., .OR., .NOT., are available in Fortran IV. These operators act on logical expressions, variables or constants to produce a logical value. Consider the statements:

```
LOGICAL B, C, D, E, F
D = B. AND. C
E = B. OR. C
F =.NOT. B
```


In the second statement, D will be given a value of true if both B and C are true, but D will be false otherwise. When the operator is .OR., E will be true if either B or C, or both, are true. The .NOT. operator will cause F to be true if B is false, or F to be false if B is true. The function of these operators is sumarized in the following truth table.

A	T	T	F	F
B	T	F	T	F
A. AND. B	T	F	F	F
A. OR. B	T	T	T	F
.NOT .A	F	F	T	T

```
L = I + Z .EQ.(J*5).OR.A**C-D.LE.R.AND..NOT.G
```

Logical operators may be combined in complicated expressions; an example is shown above. The sequence of use of operations within an expression is first, arthmetic operators, second, relational operators, and third, boolean operators. The following list shows the hierarchy of the Fortran IV operators:

```
**
unuary minus
*,/
+,-
.LT.,.LE.,.EQ.,.NE.,.GE.,.GT.
.NOT.
.AND.
.OR.
```

The major use of logical expressions is usually in the logical IF statement. This statement contains a logical expression and a complete, executable state-

ment. For example, in

```
IF (L) Y = X
Z = X
```

if L is true, Y = X will be executed and then the statement Z = X will be executed. If, however, L is false, the Y = X statement will not be used but the Z = X statement will be executed. In this example, L represents any logical variable or expression. Y = X may be replaced by almost any executable statement.

The logical IF statement has several advantages. Through the combined use of logical variables, relational operators, and boolean operators, it permits a few, more complex, expressions to replace a series of three way branches. It avoids statement numbers for impossible alternatives. Perhaps most significantly, it allows the program logic to more closely parallel the reasoning of the human programmer.

A quite different type of change from Fortran II to Fortran IV is the inclusion of similar input and output statements for all peripheral units. These Fortran IV expressions have the form

```
READ (i,n) list
WRITE (i,n) list
READ (i) list
WRITE (i) list
```


where i is a unit number which may be either an integer constant or an integer variable; n is a format number which must be a constant; and the list has the same form as in a Fortran II program. For the present 1620 system the following unit code assignments have been made:

i	READ (i,n)	WRITE (i,n)	WRITE (i) READ (i)
0	Tape 0	Tape 0	Tape 0
1	Tape 1	Tape 1	Tape 1
2	Tape 2	Tape 2	Tape 2
3	Tape 3	Tape 3	Tape 3
4	Tape 4	Tape 4	Tape 4
5	Card Reader	Tape 5	Tape 5
6	Typewriter	Typewriter	———
7	———	Card Punch	———

Input and output statements which do not include a format number cause data to be written in internal core notation. These statements allow much more rapid tape reading and writing. The unformatted input-output is often used when a magnetic tape drive is to be a temporary storage device for data which the program will use at a later time. This data transfer is so fast that it is often quicker to initialize a matrix by reading in values than by computation.

The use of a variable to specify the input or output unit allows increased program flexability. An I/O unit may be specified by a control card; this allows external control of program operation under various conditions. If the unit designation is computed internally, the output media can readily reflect special program results.

In addition to the generalized input and output statements, the Fortran IV compiler also recognizes all Fortran II card and typewriter input and output statements. Three magnetic tape control statements are also provided. These are REWIND, BACKSPACE, and END FILE.

Several minor changes have been made in the system. Execution of the STOP or END statement causes termination of the program, a search in the card reader for the next program and transfer of control to it. This feature allows batch operation of jobs. Function subprograms may have their mode specified, using a procedure similar to the specification of the mode of variables. Another innovation in Fortran IV is the possibility of declaring dimensions in a COMMON statement.

The last major change in the Fortran IV system is the inclusion of library subprograms as part of Deck III. The subprograms DUMP and PDUMP are used to obtain selective trace and for snapshot diagnosis. They differ in that PDUMP allows the program to continue while DUMP terminates it. Subprogram EXIT terminates the program in a manner similar to the execution of the END statement. Subprograms CALCMP and AXES are used for control of the Calcomp 1626-27 graph plotter.

Hardware indicators, such as sense switches and the last card indicator, are interogated in Fortran IV by the SSWTCH subprogram. This performs a similar function to, and replaces, the IF(SENSE SWITCH) statement. The switch interogated by this subprogram may be represented by a variable. Similar subprograms, SLITE and SLITET refer to simulated sense lights which are turned on and off by program control. Sense lights are often used as indicators of program status.

Although the Fortran IV system which has been described functionally has many advantages; these were achieved in part by restrictions in several areas. Precision is fixed at *0810, largely to speed program execution. The 7040 capabilities of complex and double precision data, assigned GO TO, and labeled COMMON are not available. Although their value is probably marginal, these features could be added to 1620 Fortran IV, but most would require major modifications.

The programming and development of the 1620 Fortran IV system was based on modification of existing systems, rather than the coding of a completely new system. In order to limit the cost of system development, it was decided to create Fortran IV by patching the IBM Fortran II system (1620-FO-019, version I). As a result, compilation of Fortran II statements is essentially unchanged in Fortran IV. Compilation of new statements is handled, in most cases, by slight modifications of Fortran II procedures.

The following notes on compilation procedures will serve as examples of compiler changes required. For those familiar with internal compiler operation, these notes are indicative of the major systems concepts used in Fortran IV.

1. Recognition of new statements is made possible by slight modifications of the standard scan and by the addition of new subsidiary routines where needed.
2. New 3 digit string language codes are used in several cases. Since the generalized input-output statements are given string-language values above 190, several string values in the force-table are freed for assignment to relational and boolean operators.

3. Input/output statements are all coded as the generalized input-output statements with the necessary unit code constants synthesized.
4. Relational and Boolean operators are recognized by the periods which precede and follow them. Special routines scan these statements and output appropriate string language symbols. Logical constants are interpretted by the same routines.
5. The type specification statements replace the digit in the symbol table which shows the mode of the variable. The use of the digits 0 for real and 2 for integer is retained. Logical variables and constants are assigned a code of 4.
6. Dimensions declared in type specification and COMMON statements are evaluated by the regular DIMENSION routine, at the conclusion of which control passes to the originating routine.
7. Logical IF statements are coded as three statements in the intermediate string language. The first statement is the IF, including the logical expression, and a compiler-generated, negative, statement number. The second generated statement is the conditionally-executable source statement. The third is a CONTINUE to which the negative statement number is assigned.
8. The Pass II force table and operations table have been extensively modified, although the same concepts are retained. The input-output command coding and the IF coding have been modified. Additional mode checks are also included in Pass II.

9. Subscript computation speed is increased greatly by indexing a one-dimensional array, and the first index of a multi-dimensional array, by addition rather than by multiplication.
10. After loading of the object program, a scan is made to determine which subprograms have not been loaded. If routines have not been loaded and are not on the library tape, an operator message is typed.
11. Changes in Deck III include addition of error routines for magnetic tape operation and an error check of uninitialized variables read into FAC.
12. The unformatted magnetic tape commands read into and write from core directly. The digit following the data area is saved and replaced after execution.
13. Generalized input-output commands first evaluate the unit code. Control is then transfered to specific routines for the typewriter, card reader or punch, or for magnetic tape.
14. Logical values of true and false are represented in core by their alphameric (double digit) equivalents. Logical input and output can therefore be via an A5 format. Logical branches are executed by a BD on FAC.

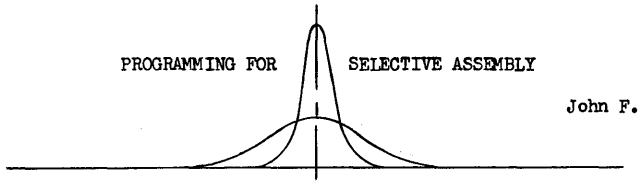
Although numerous other changes have been made, the preceding notes cover the principal ones.

In conclusion, the Fortran IV system represents a compromise which has proven highly successful. It has been used for almost all new programs written. The language is easy for a Fortran II programmer to learn; a typical instruction course might last 3 hours. Despite its increased power, it is no harder for new programmers to learn than Fortran II. Finally, the production

of this system was both interesting and instructive for its creators.

Perhaps the most significant result of the creation of this system is the demonstration that a standard 1620 Fortran IV is both feasible and useful. Although the system just explained is not a generally useful system, production of such a system would not require unreasonable effort.

More detailed information on the 1620 Fortran IV system is available from the Mayo Computer Facility, Rochester, Minnesota. Further information on the IBM 7040 Fortran IV is available in C28-6329, IBM 7040/7044 Operating System (16/32K): Fortran IV Language.



PROGRAMMING FOR SELECTIVE ASSEMBLY

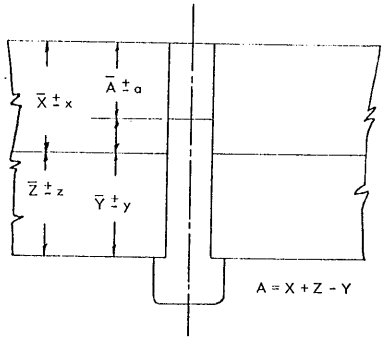
John F. Mahan

Selective assembly is a term heard in many industries. It describes the practice of matching oversized or undersized dimensions with their respective assembly counterparts to attain higher quality. In this age of interchangeable manufacture, selective assembly is often regarded as undesirable. There are occasions, however, in which the practice is the only economical answer to a problem.

The fit between piston and engine cylinders and the matching of resistors and other components in electrical devices are typical examples selective assembly operations. The existence of such examples suggests the question, "Why not use selective assembly more extensively?" The obvious answer is that such factors as one hundred percent inspection and high in-process inventory levels, promote organizational problems and high costs. A less obvious answer to the same question is that possibly planning techniques for selective assembly have not been as well developed as planning techniques for interchangeable manufacture have been developed.

This possibility led to development of the following approach to dimensional problems in assembly. The system described in this paper has been used as a classroom tool in an Industrial Engineering course, "Manufacturing Systems Design," at General Motors Institute. A program embodying the concepts described has been written for operation on the IBM 1620 computer.

The approach to the selective assembly problem, and subsequent development of an algorithm for specification of group limits will be explained in terms of a simple example.



SPECIFIED $\bar{A}$ = .250 $a = \pm .002$
CAPABILITY $\bar{X}$ = 1.250 $x = \pm .004$
CAPABILITY $\bar{Y}$ = 2.000 $y = \pm .004$
CAPABILITY $\bar{Z}$ = 1.000 $z = \pm .004$

FIGURE 1. ASSEMBLY TO BE ANALYZED IN SELECTIVE ASSEMBLY PROBLEM. CAPABILITIES GIVEN REPRESENT ± 3 STANDARD DEVIATIONS OF THE PROCESSES

The problem is exemplified in Figure 1, where the clearance (A) and 3σ limits $\pm a$ are to be held to .250$\pm$.002 respectively. Input to the assembly problem are dimensional distributions as shown in Figure 2. These distributions represent process capabilities of machines producing independent dimensions **X**, Y and Z. It is assumed that the median values of dimensions may be held to attain the specified median clearance. Dimensional relationships shown in Figure 1 may be represented by the equation:

$$A = (X + Z) - Y$$

The maximum and minimum clearances possible would then be:

$$A_{MAX} = X_{MAX} + Z_{MAX} - Y_{MIN}$$
$$= 1.254 + 1.004 - 1.996 = .262$$

$$A_{MIN} = X_{MIN} + Z_{MIN} - Y_{MAX}$$
$$= 1.246 + .996 - 2.004 = .238$$

The median dimension of .250 for A has been attained but we have violated the $\pm$.002 tolerance since if random assembly were attempted, the variation in clearance would be excessive ($\pm$.007 by summing of variances).

Selective assembly appears to be desirable unless a high scrap rate or better equipment can be tolerated by the economic system involved.

An Algorithm to Determine Group Limits

Steps of the algorithm are:

1. Expanding the variance of a distribution.
2. Plotting the clearance curve.
3. Designating the number of groups.
4. Specifying upper and lower limits on each group such that the desired clearance specifications are held.

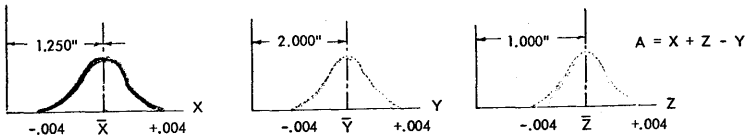


FIGURE 2. INPUT DATA TO ASSEMBLY PROBLEM (REFLECTING PROCESS CAPABILITY)

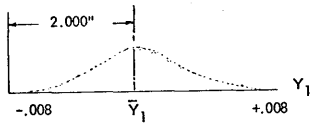


FIGURE 3. DISTRIBUTION OF DIMENSION Y AFTER MODIFICATION

5. Specifying groups so as to obtain an equal number of mating components in each group. However, the number of sets of matched components can vary from group to group, i.e., Group 1 has 20 matched sets, Group 2 may have more or less than 20, etc. Figure 5 is to be thus completed.

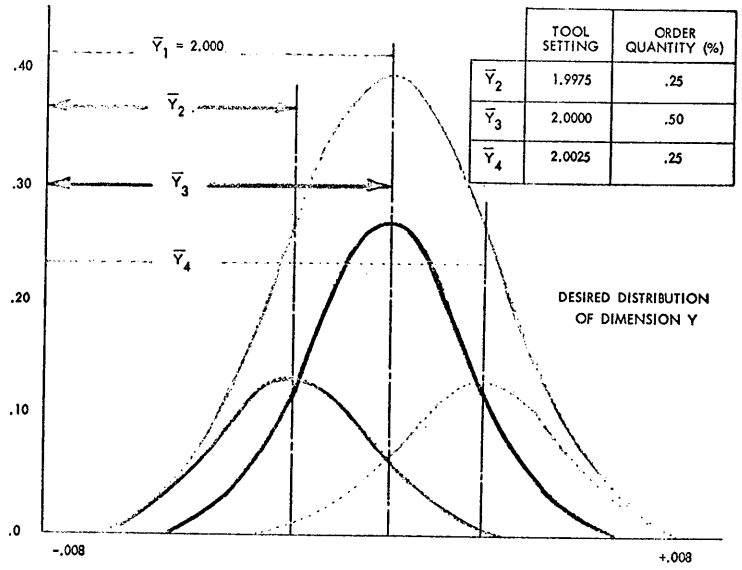
With reference to Figure 2, it may be noted that the usual heuristic technique of selective assembly is to match the smallest dimensions of X, Y and Z, thus producing the desired clearance (A) within specified tolerance (a). When the heuristic technique is used the dimension relationships first considered are:

$$A = X_{MIN} + Z_{MIN} - Y_{MIN}$$
$$= 1.246 + .996 - 1.996$$
$$= .246$$

Since this is less than the specified maximum value of .248 it becomes necessary either to "tighten" limits on X or Z or to "widen" limits on Y. Rather than attempt to tighten limits (possibly requiring extensive tooling and machining system changes) it may be desirable to expand the distribution of Y by a scheduled variation as indicated in Figure 3.

Step 1 - Expanding the Variance of a Distribution

Controlled expansion of variance may be accomplished as shown in Figure 4. It will be noted that the desired variance and distribution may be obtained by proper specification of tool settings (Y_2, Y_3, Y_4. . . . .) and quantities to be produced at these settings (areas under respective curves). This aspect of the problem has been done using an analog simulation since the analog computer is well suited to this type of work. Tool settings



	TOOL SETTING	ORDER QUANTITY (%)
$\bar{Y}_2$	1.9975	.25
$\bar{Y}_3$	2.0000	.50
$\bar{Y}_4$	2.0025	.25

FIGURE 4. PLOT OF DISTRIBUTIONS WHOSE SUM APPROXIMATES THE DESIRED DISTRIBUTION, WITH MACHINING SCHEDULE REQUIRED TO PRODUCE DESIRED DISTRIBUTION

and quantities to be produced at each setting to produce the desired distribution may be readily derived by proper scaling and iteration (details of this step are not included in this paper). Modified input to the system now consists of the distribution shown in Figure 3.

Step 2 - Plotting the Clearance Curve

The initial step toward plotting the clearance curve is to plot the cumulative distributions of each dimension (X, Y, and Z) using a common probability ordinate (Figure 6). The relationships plotted are: $F(X) = P_r(X \leq x)$, $F(Z) = P_r(Z \leq z)$, and $F(Y) = P_r(Y \leq y)$. These curves are to be used in determining the clearance curve: $F(A) = P_r(A \leq a)$. The clearance curve, F(A), represents the cumulative distribution of clearances attained if each assembly were produced so as to cause maximum variation in clearance. The F(Y) curve is actually an approximation to the function: $-(-F\text{¢}(Y))$. Figure 4b illustrates the basis for the approximation. Manipulation of the cumulative distribution as indicated simplifies subsequent steps in the algorithm. Point P on the clearance curve may then be interpreted as follows: If all dimensions X of 1.254" or greater and all dimensions Z of 1.004 or greater were assembled with all dimensions Y of 2.008 or less, then maximum possible clearance of all assemblies would be .266 or less. Similarly, the minimum possible clearance would be .234. If selective assembly is not done, these are the worst possible assembly conditions which can occur. The algorithm will specify how to avoid them. It may be noted that abscissa of any point P of the clearance curve is determined by summing abscissa values of X, Y and Z curves, i.e.,

$$a = x + z + y$$

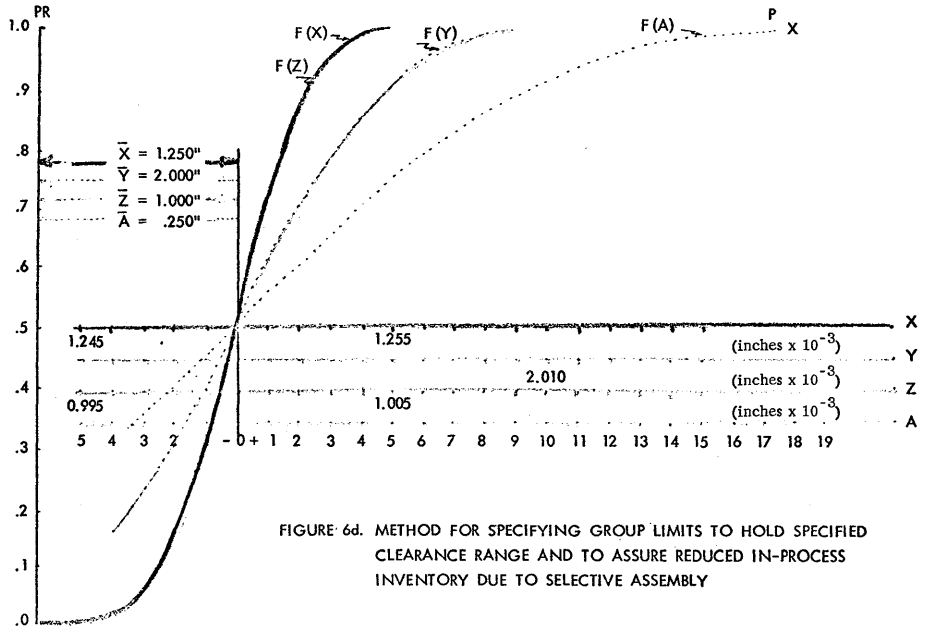
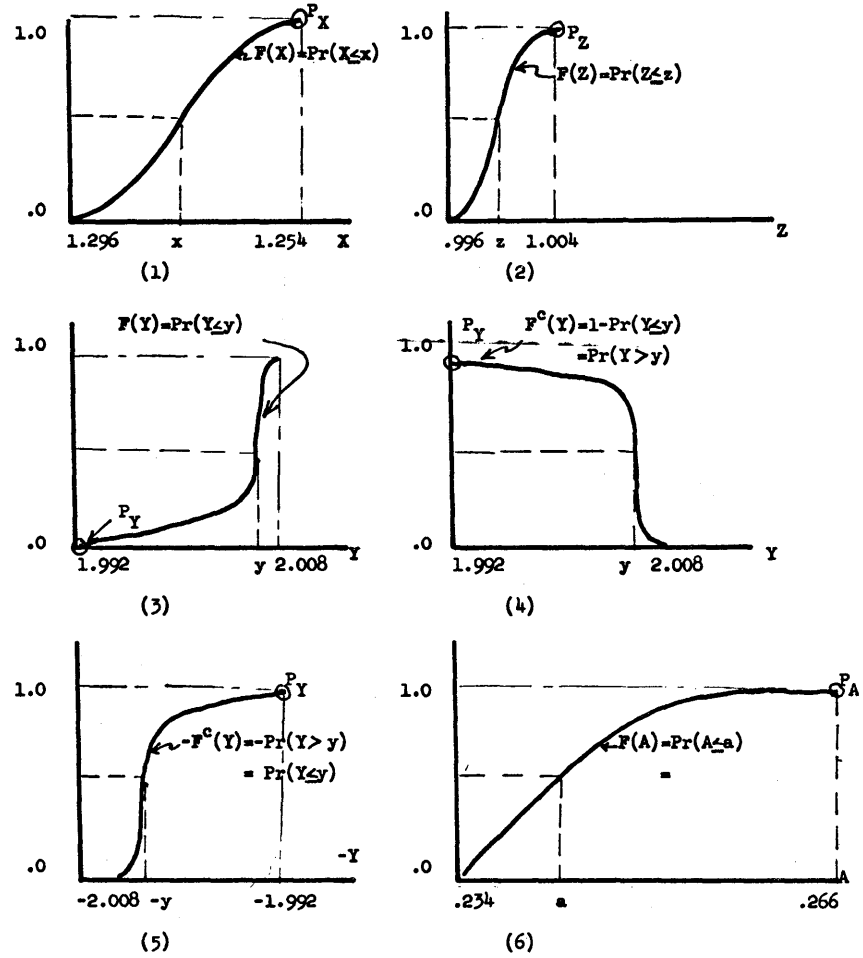


FIGURE 6d. METHOD FOR SPECIFYING GROUP LIMITS TO HOLD SPECIFIED CLEARANCE RANGE AND TO ASSURE REDUCED IN-PROCESS INVENTORY DUE TO SELECTIVE ASSEMBLY



$$A=F(X)^{-1} + F(Z)^{-1} - (-F^C(Y))^{-1}$$

$$\approx F(X)^{-1} + F(Z)^{-1} + F(Y)^{-1}$$

FIGURE 4b STEPS ILLUSTRATING THE BASIS FOR THE ASSUMPTION: $F(Y) \approx -(-F^C(Y))$. NOTE THAT SYMMETRY OF DISTRIBUTION Y DETERMINES THE ACCURACY OF THE ASSUMPTION.

The clearance curve may then be plotted by summing the abscissa as indicated for each common ordinate value.

Step 3 - Designating the Number of Groups

Since the desired clearance range (a) is ±.002 and the maximum clearance range possible is ±.016 (between .234 and 2.66) the number of groups (N) required would be:

$$N = \frac{\text{Actual Range}}{\text{Desired Range}} = \frac{.016}{.002} = 8$$

This assumption does not allow for random assembly within the individual groups but is an acceptable solution. This aspect will be explained subsequently.

Step 4 - Specifying Limits for Each Group

Specification of the lower limits for the first group is accomplished as shown in Figure 6e. The lower limits on X, Y and Z may be determined by begining at the abscissa value, equal to the nominal value of X, Y and Z. These values are entered in Figure 5b. Upper limits for group 1 are determined by indexing on the clearance scale for a distance equal to the desired range (2a or .004 in the example).

A trace is then drawn upward to the clearance curve, thence horizontally to intersect the cumulative curves F(X), F(Y), and - F^C(Z) and downward to the respective X, Y or Z abscissas. The upper limit on each dimension may be entered in Figure 5c. These upper limits become the lower limits for the following group.

Upper limits for Group 2 are obtained similarly by reference to the .008 value on the clearance abscissa. Subsequent group limits are determined by indexing across the clearance abscissa in increments of 0.004.

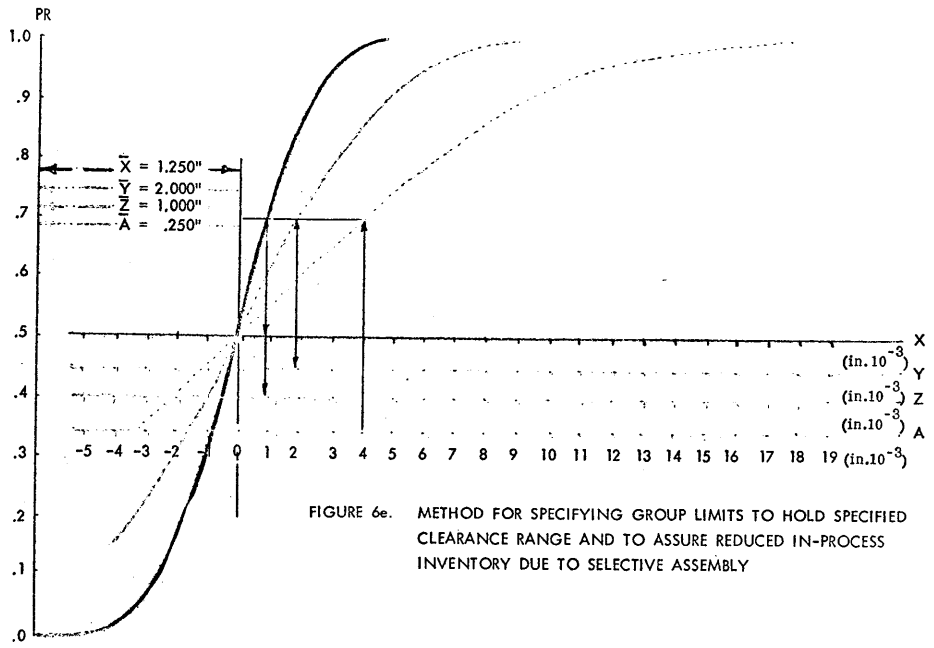


FIGURE 6e. METHOD FOR SPECIFYING GROUP LIMITS TO HOLD SPECIFIED CLEARANCE RANGE AND TO ASSURE REDUCED IN-PROCESS INVENTORY DUE TO SELECTIVE ASSEMBLY

GROUPS

		1	2	3	4	5
X	U					
	L	1.2500				
Y	U					
	L	2.0000				
Z	U					
	L	1.0000				
A	U					
	L					
0/0 PARTS IN GROUP						

FIGURE 5b

TABLE TO BE USED IN CLASSIFICATION OF ASSEMBLY COMPONENTS SO AS TO HOLD SPECIFIED CLEARANCES AND REQUIRE LOW INVENTORY FOR IN-PROCESS PARTS

GROUPS

		1	2	3	4	5
X	U	1.2509				
	L	1.2500				
Y	U	2.0019				
	L	2.0000				
Z	U	1.0009				
	L	1.0000				
A	U	.2518				
	L	.2481				
0/0 PARTS IN GROUP		19.5				

FIGURE 5c

TABLE TO BE USED IN CLASSIFICATION OF ASSEMBLY COMPONENTS SO AS TO HOLD SPECIFIED CLEARANCES AND REQUIRE LOW INVENTORY FOR IN-PROCESS PARTS

GROUPS

		1	2	3	4	5
X	U	1.2509	1.2519			
	L	1.2500	1.2509			
Y	U	2.0019	2.0038			
	L	2.0000	2.0019			
Z	U	1.0009	1.0019			
	L	1.0000	1.0009			
A	U	.2518	.2519			
	L	.2481	.2480			
0/0 PARTS IN GROUP		19.5	15.2			

FIGURE 5e

TABLE TO BE USED IN CLASSIFICATION OF ASSEMBLY COMPONENTS SO AS TO HOLD SPECIFIED CLEARANCES AND REQUIRE LOW INVENTORY FOR IN-PROCESS PARTS

Step 5 - Estimating Number of Parts in Each Group

The estimated percentage of assemblies which will be in each group may be determined by noting the difference in the ordinate values between the respective horizontal traces used in determining the group limits. Since Group 5 will contain less than one percent it is not shown. Group limits to the left of the median are also not shown, but may be similarly determined. It may now be noted that the desired clearance, $A \pm a$, within each group has been held and falls within .248 and .252. Any desired tolerance on the clearance may be held by proper specification of the index on the clearance scale.

Output from the system (Figure 5f) is now complete and may be evaluated by direct application, or by digital simulation to determine in-process inventory levels required by the selective assembly program shown in Figure 5f.

The in-process inventory may also be deterministically estimated, but the method becomes quite lengthy. For example, the probability of the first randomly selected X, Y and Z dimensions falling into Group 1 may be estimated by noting:

$$P_r\,(1.2500 < X < 1.2509) = P_r\,(2.0000 < Y < 2.0019)$$

$$= P_r\,(1.0000 < Z < 1.0009) = .195$$

and their joint probability would equal their products or $(.195)^3$. Similarly, the probability of the initial assembly being acceptable (all three dimensions falling into any one group) would be:

$$P_r(\text{acceptable assembly, first draw}) = \sum_{i=1}^{N} P_i$$

where P_i = percent of parts in each individual group

and N = number of groups

GROUPS

		1	2	3	4	5
X	U	1.2509	1.2519	1.2529	1.2536	
	L	1.2500	1.2509	1.2519	1.2529	
Y	U	2.0019	2.0038	2.0059	2.0073	
	L	2.0000	2.0019	2.0038	2.0059	
Z	U	1.0009	1.0019	1.0029	1.0036	
	L	1.0000	1.0009	1.0019	1.0029	
A	U	.2518	.2519	.2520	.2513	
	L	.2481	.2480	.2479	.2485	
0/0 PARTS IN GROUP		19.5	15.2	10.8	4.3	0.2

FIGURE 5f

TABLE TO BE USED IN CLASSIFICATION OF ASSEMBLY COMPONENTS SO AS TO HOLD SPECIFIED CLEARANCES AND REQUIRE LOW INVENTORY FOR IN-PROCESS PARTS

In the example:

P_r(acceptable assembly, first draw)

$= (.195)^3 + (.152)^3 + (.108)^3 + (.043)^3 + (.2)^3$

$= .0123$ or for the entire distribution

$P_r = (.0123)(2) = .0246$

Determing the probability of subsequently drawn assembly components falling into mating groups requires the use of conditional probabilities which will not be discussed here.

Summary

The approach to the selective assembly problem has been as follows:

1. The clearance equation is written in terms of the related dimensions ($A = X + Z - Y$).
2. Median values for related dimensions are adjusted to satisfy the dimensional equation ($\bar{A} = (\bar{X} + \bar{Z}) - \bar{Y}$).
3. Distributions of related dimensions are analyzed and if necessary, variances may be expanded by simulation.
4. Cumulative distributions F(X), F(Y), F(Z) are plotted according to clearance equation relationships with a common probability ordinate.
5. The extreme clearance curve, F(A), is plotted according to clearance equation relationships (using a common probability ordinate).
6. Using an arbitrary point on the clerance abscissa and an index equal to the desired range, (2a), upper and lower limits for each group may be obtained by transformation of abscissa values to probabilities which are then transformed to respective limits on related dimensions (X, Y and Z).

In conclusion, it should be noted that the algorithm described functions satisfactorily, but is non-optimal in the area of the required number of groups. Random assembly of parts within the groups will reduce the number of groups required. The final evaluation of any proposed selective assembly program, however, must consider inventory levels versus quality and costs. While the above factors may be evaluated by a subsequent Monte Carlo simulation, the best program evaluation will be made in the field.

EXPERIENCES WITH "MISS-LESS" AND "MISS-LESS DATING"

A Presentation for the 1620 Users Group
Fall Meeting
Norman, Oklahoma
November 10, 1964

TRUNKLINE GAS COMPANY
Engineering Department
Engineering Planning

L. V. PARENT

R. E. DILLON

EXPERIENCE WITH MISS-LESS & MISS-LESS DATING SYSTEMS

INTRODUCTION

Trunkline Gas Company

Trunkline Gas Company is an interstate gas transmission company, with a Federal Power Commission certificated capacity of 890 million cubic feet per day. It has roughly 160,000 horsepower installed in 15 compressor stations located along its main pipe line which runs from McAllen, Texas, to Elkhart, Indiana. Total length of all of the company's pipeline is about 2700 miles. In the summer of 1964, two new compressor stations were built, and additions to four stations were installed, at a total cost of about $9,000,00. The project was typical of the ones pursued almost annually since 1960 to supply the demand for natural gas in the Mid-West.

Management of the Project

Trunkline, when installing new facilities, performs the functions of design, construction inspection, and materials procurement, but retains an independent contractor to do construction work. A project engineer is assigned to each job. He is the one responsible for:

Designing facilities which will perform as required

Completing construction by the time scheduled

Keeping costs within the construction budget (1).

The engineer has the authority and the direct means of using it to discharge his first responsibility, because design is a company function. His success in discharging the other responsibilities is strongly affected by the rigidly defined terms of the construction contract. Failure to provide design data, or construction material becomes a matter not for inter-departmental complaint, but the source of a costly contractual dispute. Prudence requires that the means given him in the contract---the right to take over the job---be seldom, if ever, used. Satisfactory accomplishment of the construction plan must, therefore, be attained by less direct means. The engineer first anticipates, or defines a problem, then persuades the contractor of its importance, and often diplomatically guides its solution. Procedures for project planning and progress reporting are his necessary aids, and their importance is recognized by the project management. After each facilities expansion, project planning, scheduling, and progress reporting-recording techniques are reviewed, then revisions are made in a continuing effort to improve effectiveness.

USING CRITICAL PATH METHODS

Introduction of the Critical Path Method

When the 1961 projects were studied much introductory level information on PERT and CPM could be obtained, and the freedom to choose a procedure was widened because a small paper tape input computer was available. After review, the Critical Path Method (CPM) was chosen as the basis of a procedure for project planning and progress reporting for the 1963 construction program. It was revised as necessary while the

construction program went on. Graphical analysis of the arrow diagram supplanted computer analysis. The computer program was ineffective since it had restrictions on job numbering and could not generate calendar dates.

The scheme as it finally evolved is outlined, following:

Data Gathering

The Bid Invitation letter included instructions that the "bidder shall submit a construction progress schedule, and an activity oriented arrow diagram which can be used ... in critical path method analysis of the schedule." If the bidders asked questions about CPM they were referred to "A Special Report-Critical Path Scheduling" (2). Proposal review clearly showed some of the contractors were taking their first lesson in CPM. The project engineer added material, operational, or other restraints to the contractor's diagram to completely delineate the logic of the project.

Plan Analysis

The completed arrow diagram was then analyzed. This process required the efforts of an engineer and draftsman, who, following the rules for a graphical method established in a paper by C. W. Lowe, (3) found the critical path, free float and calendar dates. The resulting plot was called a "Lowe diagram" to distinguish it from a conventional arrow diagram. Engineers and draftsmen were notably unenthusiastic about this part of the procedure.

Schedule Preparation and Progress Reporting

The recommendation, made in most of the CPM literature, that the arrow diagramming and scheduling parts of the method be rigidly segregated was ignored in this part of the procedure. Project management wanted a document that would be as detailed as an arrow diagram, but which would show the time relations of a bar chart. The chart - "Construction Plan & Schedule," (Exhibit I) - became the project manager's primary graphic record. It is essentially an arrow diagram plotted to a calendar scale, each job being shown at its earliest conveniences.

The diagram was satisfactory as long as deliveries were made when promised, and work was done near the dates predicted. On some jobs, slippages occurred; the chart continued in use but became progressively more unrealistic. Time to make the major revisions which would have showed a current prediction of project course was never available.

A Revised Procedure

The Trunkline IBM 1620 was delivered in April 1964, just as another construction program commenced. MISS-LESS and MISS-LESS DATING were studied, and a procedure, based on the two programs was worked out.

Data Gathering

Information from the contractor was received as before, and the arrow diagram was completed by the project engineer. A material clerk then took off the node numbers, durations, and activity names using a form (Exhibit II). Cards were punched by Accounting Tabulating.

Plan Analysis

MISS-LESS and MISS-LESS DATING were run by the clerk to produce a deck for listing on a 1403 printer. The project engineers found the 1620 program print-out (Exhibit III) difficult to interpret. Accounting Tabulating wrote a SPS program for the 1401-1403 which produces the print-out shown in Exhibit IV. The format revision was made to encourage engineer analysis of the plan before the charts were plotted, but did not achieve its purpose since the engineers continued to be chart viewers rather than list readers.

Schedule Preparation and Progress Reporting

A draftsman was given the printout. He was able quickly, and with little supervision to plot the "Construction Plan and Schedule." Slippages of all varieties occurred during the project. The problems of updating, replanning, and rescheduling on a difficult-to-revise chart became insurmountable. When schedule delinquencies became too great, use of the charts simply ceased.

The procedure followed in 1964 had both faults, and virtues. Easy revision, and graphical representation, as in 1963, were problems. Computer calculation replaced graphical calculation, project engineers were spared some tedious work, and drafting time was reduced.

MAKING CPM MORE EFFECTIVE

The capabilities of the 1620-1311 Computer and the 1401-1403 make it possible to propose the design of a CPM procedure which not only would lack the defects of 1963 and 1964 procedures, but which would also include other features. There have been two years of experiment, so the questions asked the designer of this system by the project managers will be searching. The designer will first be required to make assurances that the already heavy load of procedural work will not be increased. He will then probably have to answer the following questions about the system in the affirmative. (4).

Does it help me plan the course of the project?

Is the plan presented graphically?

Does it show me the exceptions to the plan?

Is it easily revised?

The first question can be answered affirmatively because MISS-LESS and MISS-LESS DATING are available.

An adequate graphical representation will have to be designed. It is of primary importance since engineers and managers usually reject lists, or tabulations. The right answer to this question will probably be found with difficulty. A good system would show all exceptions to plan. Some project engineers look only at exceptions on the critical path through the plan, others consider any exception to plan an important matter. The Programs for the 1401-1403 which print diagrams might solve these twin problems of graphical representation and exception indication.

If the system planner programs the 1620 and 1401 properly he will almost certainly be able to answer the last question correctly. Revision should only be a matter of card handling, and computer operation.

Each of these questions springs from a need the project engineer-user wants met. Programs to give the answers may already have been written for specific projects, if so, their generalization, and distribution would be helpful. If not, then a series of programs to satisfy the wants of project management should be prepared. Trunkline's experience shows this is the next step if the Critical Path Method is to become as useful as it can be.

REFERENCES

1. Paul O. Gaddis, "The Project Manager," Harvard Business Review, Vol. 37, No. 3. (May-June 1959) 89-97.

2. "A Special Report - Critical Path Scheduling," Hydrocarbon Processing and the Petroleum Refiner, Vol. 41, No. 2 (February 1962) 123-138.

3. C. W. Lowe, "Critical Path Scheduling Simplified," Chemical Engineering (December 10, 1962) 170-176.

4. Paul E. Hall, Presentation at CIMM-AIMME Petroleum Engineering Section, Calgary, Alberta, Canada, May 13, 1958.

ACKNOWLEDGEMENTS

The help of Mr. R. C. Newsome, Manager of Engineering Design, and Mr. J. B. Sellers, Supervising Engineer, Engineering Design, who permitted the use of exhibits and who contributed much to the expression of the project engineer's viewpoint is acknowledged.

WRITING FORTRAN II (AND FORTRAN II-D) SUBROUTINE SUBPROGRAMS IN SPS FOR THE IBM 1620 COMPUTER

R. J. Wepman

IBM Data Systems Division

ABSTRACT

This writeup provides complete instructions for the coding, assembling and utilizing of SPS written subroutine and function subprograms in 1620 FORTRAN II or FORTRAN II-D. A knowledge of the 1620 SPS II and FORTRAN II systems is presupposed.

BACKGROUND

Ideally, a compatible assembler, such as FAP, for the IBM 7090, should be provided to the user of a system for which FORTRAN II has been written. In the case of the IBM 1620, this was not done for non-disk systems.

It is possible, however, to use 1620 SPS II or SPS II-D for this purpose and, in the case of SPS II programs, to perform some manual manipulations with the object deck to make it compatible with the FORTRAN II systems.

PHILOSOPHY OF OPERATIONS

Both 1620 FORTRAN II and FORTRAN II-D utilize what is known as a relocating loader. This means that the subprogram being loaded is shifted, in memory, to an area not yet occupied by any of the programs previously loaded by the system. The loader is instructed to relocate (or not) by the presence (or absence) of flag operands in positions O_0 and O_1, in the case of instructions, and by suitable characters punched in certain columns of object cards produced by DSA and DC mneumonics.

CALL AND FUNCTION LINKAGES

When subroutine subprograms are called from a FORTRAN II program; e.g.,

```
CALL SUB (ARG 1, ARG 2, ..... ARG N)
```

or function subprograms are requested; e.g.,

```
Y = FOF (X, Y)
```

FORTRAN II assembles the following sequence of instructions:

```
BTM     - SUBAD, *+11
DSA     ARG 1, ARG 2, ..... ARG N
```

SUBAD is the address in memory where the entry address of subroutine subprogram "SUB" has been placed by the loader. Similarly, the function request is of the form:

```
BTM     - FOFAD, *+11
DSA     X, Y
```

i.e., they are identical in structure. The only operational difference is that, at some point in its operation, the function subprogram must place the desired result (or zero, if no computational result is required or generated) in the pseudo accumulator (symbolic name FAC in the program listings).

For both systems, the first linkage (first reference to the subprogram) structure is the same. However, the loader, in certain instances, is instructed to insert the actual entry address of the subroutine or function subprogram into the BTM instruction.

DIMENSIONED VARIABLES IN ARGUMENT LISTS

If X is an array in memory, and X appears as an argument in a CALL statement or in a function request, philosophically different linkages are generated by FORTRAN II and FORTRAN II-D.

In FORTRAN II, the address placed in the DSA and, hence, the address transmitted to the subprogram is, in effect, the address of the zeroth element of the array. This means that the address must be incremented by (FF+2) to obtain X(1), by 2(FF+2) to obtain X(2) ...., by J(FF+2) to obtain X(J).

In FORTRAN II-D, the address transmitted to the subprogram is that of the first element in the array. Thus, the J^{th} element would be obtained by incrementing the transmitted address by (J-1) (FF+2).

If the array is fixed point, the J^{th} elements would be found by incrementing the transmitted address by J(KK) and (J-1)KK respectively.

The extension of this discussion to two and three dimensional arrays is straightforward.

VARIABLES IN COMMON STORAGE

Variables are placed in common storage for two reasons:

1. to shorten and simplify the argument lists in SUBROUTINE and CALL statements;
2. to decrease the running time of SUBROUTINE subprograms.

The decrease in running time obtained by placing variables in common storage is due to the fact that the compiler is able to compute the actual addresses of the variables appearing in the argument list during compile time.

Variables in COMMON are assigned addresses from the top of memory down; i.e., the first variable (assuming it is not an array) in the COMMON statement is assigned the address n9999, (n=1, 3 or 5).

If a variable name in the COMMON statement is the name of an array, then the next available address in common is assigned to the last element of the array, and sequentially lower numbered addresses are assigned to the remainder of the array.

For example: The following statements would generate the storage assignments shown

```
        *0805           FORTRAN II
or      *FANDK0805      FORTRAN II-D
        DIMENSION       X(5), Y(3)
        COMMON          A, I, B, X, J, Y
```

Storage

Name	Address
A	n9999
I	n9989
B	n9984
X(5)	n9974
X(4)	n9964
X(3)	n9954
X(2)	n9944
X(1)	n9934
J	n9924
Y(3)	n9919
Y(2)	n9909
Y(1)	n9899

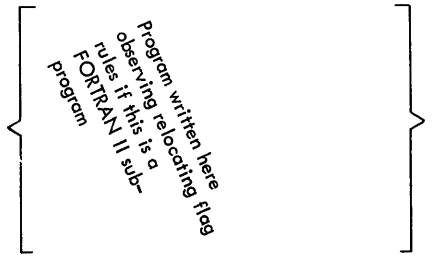
THE ACTUAL SUBPROGRAM

The first thing to be done in the subprogram is to get the actual addresses of all the variables in the argument list and the return address to the calling program.

FORTRAN II

Calling Sequence: CALL SUB (ARG1, ARG2, ... ARG N)

```
Subroutine          DORG    SUBSTART
        ARG 1       DS      5
        ARG 2       DS      5
          .           .       .
          .           .       .
          .           .       .
        ARG N       DS      5
                    DS      7
        START       AM      START-1, 5, 010,
                    TF      ARG 1, START-1, 0111,
                    BNF     *+36, ARG 1, 01,        } Get ARG 1
                    CF      ARG 1,,0,
                    TF      ARG 1, ARG 1, 0111,
                    AM      START-1, 5, 010,
                    TF      ARG 2, START-1, 0111,
                    BNF     *+36, ARG 2, 01,        } Get ARG 2
                    CF      ARG 2,,0,
                    TF      ARG 2, ARG 2, 0111,
                     .       .
                     .       .
                     .       .
                    AM      START-1, 5, 010,
                    TF      ARG N, START-1, 0111,
                    BNF     *+36, ARG N, 01,        } Get ARG N
                    CF      ARG N,,0,
                    TF      ARG N, ARG N, 0111,
                    AM      START-1, S, 010,        } Get Return Address
```




```
                    B       START-1,,06, RETURN STATEMENT
                    DEND
```

where:

S = 1 if N is even

S = 2 if N is odd

SUBSTART* = 12000 + r * 5

and r = number of library function subroutines in the system.

*assuming the system has not been re-originned

The procedure for FORTRAN II-D is covered to an understandable degree in the Monitor I write-up (C26-5739).

RULES FOR FORTRAN II SUBPROGRAMS

In order for the FORTRAN II loader to find the correct entry address and to allot sufficient space for the subprogram, the following two rules must be followed:

1. The first number storing statement of the subprogram must be an instruction. A number storing statement is one which causes actual numeric information, flags, etc., to be punched into an object card, as opposed to those which simply advance the address counter. Examples of number storing statements are:

   a) DC

   b) DAC

   c) DSA

   d) DNB

   .

   .

   .

   and instructions

   Non-number storing statements are:

   a) DS

   b) DSB

   c) DORG

2. The last statement (aside from the DEND operand) of the subprogram must be a number storing statement, preferably an instruction.

In FORTRAN II-D, the above restrictions are unnecessary since both the entry address and length of program are available to the loader.

FLAG OPERANDS FOR FORTRAN II

If P is relocatable, flag operand is 0

If Q is relocatable, flag operand is 1

If P and Q are relocatable, flag operand is 01

If P and Q are relocatable and P is indirect, flag operand is 016

If P and Q are relocatable and Q is indirect, flag operand is 0111

If P is relocatable and Q is immediate non-relocatable, flag operand is 07, 08, 09, 010 or 011 depending upon where the immediate flag is to be set

If P is relocatable and Q is immediate relocatable, flag operand is 017, 018, 019, 0110 or 0111 depending upon where the immediate flag is set

If P is not relocatable and Q is immediate non-relocatable, flag operand is 7, 8, 9, 10 or 11

If P is not relocatable and Q is immediate relocatable, flag operand is 17, 18, 19, 110 or 111

If neither P nor Q are relocatable, no flag operand is employed

If P and Q are relocatable and indirect, flag operand is 01611

CALLING LIBRARY FUNCTIONS FROM SPS SU PROGRAMS

In FORTRAN II, a 50-digit record beginning in location $\bar{0}$0415 thru $\bar{0}$0464 is used as an indicator record to tell it which library functions are required for proper execution of the program being loaded. The indication is done with a sequence of ones and zeros. A one in location $\bar{0}$0414 + I means that the function numbered I is to be loaded, and its address is to be stored in location ǂ11999 + 5(I). The value I is determined by the order in which the FUNCTION has been added to the library deck and to the PASS I deck.

A zero in position $\bar{0}$0414 + I causes the I^{th} function subprogram to be ignored. The loader is instructed to load ones into the indicator record by a card punched in the following format (FORTRAN II only):

COLS	CONTAINS
1	1
2	≠ (0-2-8 punch)
3 - 61	Blank
62	Flag 1 (J punch)
63 - 64	Blank
65 - 69	$\bar{0}$0414 + I (Note High Order Flag)
70 - 74	$\bar{0}$0415 + I (Note High Order Flag)
75 - 80	Sequence Number

ǂ assuming system has not been re-originned.

In FORTRAN II-D, the loading of a library function subprogram is requested by placing ones in the proper position of the header record as described in the manual. The entry addresses will be stored as shown in the system listings. The linkage to the library function subprogram is adequately discussed in the writeups for the respective systems.

CALLING OTHER SUBROUTINES FROM THE SPS SUBPROGRAMS

To call another subroutine subprogram from an SPS subprogram, it is necessary to add subroutine call cards (FORTRAN II only) to the object deck. These cards have the following format:

COLS	CONTAINS
1 - 12	Name of subroutine being called, in two digit coding, with a flag over high order zone digit
13 - 17	*Address where entry address of subroutine is to be stored
18	≠ (0-2-8 punch)
19 - 61	Blank
62	≠ (0-2-8 punch)
63 - 74	Blank
75 - 80	Sequence Number

*This is the address in the calling program prior to relocation. To obtain this number, it will be necessary to assign the address a symbolic name and then read off the numeric value from the map typed out at the end of PASS II.

For FORTRAN II-D, the technique shown in the manual is correct, with one rather vague point in need of discussion, since corrected.

If the SPS subprogram does not call any other subprograms, the first even location following the end of the program must contain a record mark. This is easily done by making the last statements

```
        B       ENTRY ADDRESS - 1, 0, 06,
LAST    DC      1, @
        DEND
```


This will accomplish both the insertion of the record mark and the determination of the length of the program.

Another point to be noted is that a subroutine call card should appear only once for each subroutine requested. All other linkages to the same subroutine should reference (indirect) the location in which the entry address is stored. Also, the entry address does not have to be stored in the BTM instruction, as implied in the Monitor I write-up, (although it is most economical) but may be stored in any location within the confines of the program.

THE HEADER RECORDS

The header record for FORTRAN II-D can be assembled as shown in the Monitor I manual. However, two notable changes should be made to guarantee 100% agreement between FORTRAN II-D compiled subprograms and SPS II-D subprograms. These will result in header record statements as shown:

```
S       DS      ,*+101
        DC      6,987898, 5-S
        DAC     6,NAMEbb, 7-S
        DVLC    22-S, 5, LAST, 2, ff, 2, kk, 5
                ENTRY ADDRESS -6, 5, NEXT COMMON, 30,0
        DSC     17, 01234567891234567,0
        DORG    S-100
```

However, in FORTRAN II-D, with many functions requested from the library, it may be physically impossible to make the DVLC fit on one card. Should this be the case, it will be necessary to manually duplicate the header card and hand punch the 1's into the appropriate columns.

The FORTRAN II header record is as follows:

COLS	CONTAINS
1 - 12	Subroutine name in two-digit coding, right justified, with a flag over the high order zone digit
13 - 20	Blank
21 - 22	FF Flag over high order digit
23 - 24	KK Flag over high order digit
25 - 62	Blank
63	≠ (0-2-8 punch)
64 - 74	Blank
75 - 80	Sequence Number $\bar{0}00001$

In addition to a header card, the FORTRAN II subprogram requires a trailer card (at the end of the deck) of the following form:

COLS	CONTAINS
1 - 62	Blank
63	Flagged 1 (J punch)
64 - 74	Blank
75 - 80	Sequence Number

ASSEMBLY OF SUBPROGRAM FOR FORTRAN II

1. Assemble and compress program using SPS II.
2. Remove and destroy first two cards (loader) of object deck, and the last seven cards (tables).
3. Add header record (first).
4. Add all function and subroutine call cards following last card of object deck. Sequence them accordingly. (Or use sequencing program described in appendix.)
5. Add trailer record (last).
6. Punch a flagged zero (11 zone punch) in col 62 of cards produced by DSA statements.
7. Punch a flagged 1 in col 62 of all non-relocated constants.

The subroutine is now ready for inclusion in the program stack for loading.

UTILIZING THE FORTRAN II INTERNAL ARITHMETIC AND I/O SUBPROGRAMS

The following section describes, in detail, the fixed routines which are always in memory when the FORTRAN II system is being used:

1. Name: FIX
   Linkage: BTM FIX, 0, 10
   Operation: Floating point data in FAC is converted to fixed point integer of KK digits
   Error: Error print out if integer portion floating point number is greater than 10^{kk}
2. Name: FXA
   Linkage: BTM FXA, B
   Operation: Fixed point data ot B is added to the contents of FAC
   Error: Error print out if overflow occurs
3. Name: FXS
   Linkage: BTM FXS, B
   Operation: Fixed point data at B is subtracted from the contents of FAC
   Error: See FXA
4. Name: FXSR
   Linkage: BTM FXSR, B
   Operation: Fixed point data in FAC is subtracted from B. Results placed on FAC
   Error: See FXA

5. Name: FXM

Linkage: BTM FXM, B

Operation: Fixed point data at B is multiplied by contents of FAC. Results placed in FAC

Error: See FXA

6. Name: FXD

Linkage: BTM FXD, B

Operation: Fixed point data in FAC divided by B. Result placed in FAC

Error: See FXA

7. Name: FXDR

Linkage: BTM FXDR, B

Operation: Fixed point data at B is divided by contents of FAC. Result placed in FAC

Error: See FXA

8. Name: RSGN

Linkage: BTM RSGN, 0, 10

Operation: Sign on either floating or fixed point data in FAC is reversed

Error: None

9. Name: FLOAT

Linkage: BTM FLOAT, 0, 10

Operation: Fixed point integer in FAC is converted to a floating point number

Error: None

10. Name: FSB

Floating point version of FXS

11. Name: FAD

Floating point version of FXA

12. Name: FSBR

Floating point version of FXSR

13. Name: FMP

Floating point version of FXM

14. Name: FD

Floating point version of FXD

15. Name: FDVR

Floating point version of FXDR

16. Name: TOFAC

Linkage: BTM TOFAC, X

Operation: Contents of X loaded into FAC (X either fixed or floating point)

Error: None

17. Name: FRMFAC

Linkage: BTM FRMFAC, X

Operation: Contents of FAC placed in X (X either fixed or floating point)

Error: None

18. Name: ZERFAC

Linkage: BTM ZERFAC, 0, 10

Operation: FAC set equal to floating point zero

THE I/O ROUTINES

The FORMAT statement specifications are converted, by FORTRAN II, into a string of numeric digits in one of three forms:

a) ADDRESS, W, D for E and F type specifications. (Ew.d or Fw.d)

b) ADDRESS, 2 × W for I, A, H and X type specifications. (Aw, Iw, wX, wH)

c) ADDRESS, for Slash operand. (/)

Where ADDRESS is the actual address, as found in the listing, of one of the symbolic names shown below:

ETYPE

FTYPE

ATYPE

HTYPE

ITYPE

XTYPE

SLASH

and W is a 3-digit constant ($\overline{X}XX$); D is a 2-digit constant ($\overline{X}X$).

For example: If we let

$\overline{E}EEEE$ be the actual address of symbol ETYPE

$\overline{F}FFFF$ be the actual address of symbol FTYPE

etc.....

then the format specifications F5.1, 3X, F9.2, E10.6, I4, A4 would be stored in memory as follows:

$\overline{F}FFFF\overline{0}05\overline{0}1\overline{X}XXXX\overline{0}06\overline{F}FFFF\overline{0}09\overline{0}2\overline{E}EEEE\overline{0}10\overline{0}6\overline{I}IIII\overline{0}08\overline{A}AAAA\overline{0}08$

In the case of the H specification, the Hollerith information, in the 1620 double-digit coding, follows the three-digit W constant; i.e.,

2HAB yields $\overline{H}HHHH\overline{0}04\underbrace{\overline{4}1}_{A}\underbrace{\overline{4}2}_{B}$ Note High Order Flags

REPEATED FORMATS

Let $\overline{R}RRRR$ be the actual address of the symbolic variable REP, then the format specifications F3.2, nn (I4, E5.3) would generate the following string of I/O constants:

$\overline{F}FFFF\overline{0}03\overline{0}2\overline{I}III\overline{I}\overline{0}08\overline{E}EEEE\overline{0}05\overline{0}3\overline{R}RRRR\overline{Z}ZZZZ\overline{n}n$

$\overline{Z}ZZZZ$

where the $\overline{n}n$ (following $\overline{Z}ZZZZ$) is the number of repeats desired.

In words, the routine, REP, repeats to the specification whose address is $\overline{Z}ZZZZ$, $\overline{n}n$ times.

The entire FORMAT statement, when encoded by FORTRAN II, has the following structure:

(Where $SPEC_i$ is any of the previously discussed specifications; i.e., REP, ITYPE, etc.....)

```
FORMAT   B      AROUND, SPEC 1
         DC     3, W,, (if applicable)
         DC     2, D,, (if applicable)
         DSA    SPEC 2
         DC     3, W,, (if applicable)
         DC     2, D,, (if applicable)
         .
         .
         .
         DSA    SPEC N
         DC     3, W,, (if applicable)
         DC     2, D,, (if applicable)
         DSA    REDO, LAST LEFT
AROUND   BEGINNING OF NEXT INSTRUCTION SET
```

The BRANCH to AROUND permits the FORMAT statement to be placed in the instruction mainline rather than in a separate area of the program.

The last two constants, REDO and LAST LEFT, cause the automatic recycling through the FORMAT statement, should the list exceed the number of specifications. REDO can be found in the listings, and LAST LEFT is the address of the last digit of the specification preceding the last left parenthesis.

For example:

FORMAT (F5.2, F3.1, (F5.2, F5.0)) encodes as: (excluding relocation flags)

```
FORMAT    B      AROUND, FTYPE
          DC     3, 5
          DC     2, 2
          DSA    FTYPE
          DC     3, 3
LSTLFT    DC     2, 1
          DSA    FTYPE
          DC     3, 5
          DC     2, 2
          DSA    FTYPE
          DC     3, 5
          DC     2, 0
          DSA    REDO, LSTLFT
AROUND    NEXT INSTRUCTION SET
```


THE ACTUAL I/O STATEMENT

For all succeeding discussions, the FORMAT shown in the preceding example will be used.

The Alphabetic I/O operations are all of the following structure:

1) Initialize
2) Read or Write
3) Finalize

The initialize instruction is one of the following, depending on which I/O device is being used:

```
          BTM    RACD, FORMAT +6
or        BTM    RATY, FORMAT +6
or        BTM    RAPT, FORMAT +6
or        BTM    WACD, FORMAT +6
or        BTM    WATY, FORMAT +6
or        BTM    WAPT, FORMAT +6
```

The read or write instructions are always:

```
          BTM    SWC, VAR 1
          BTM    SWC, VAR 2
           .      .    .
           .      .    .
           .      .    .
          BTM    SWC, VAR N
```

If VAR 1 is a fixed point variable, a flag should be set at Q_{11}*, no flag should be set if VAR 1 is floating point.

The I/O operation is concluded by the instruction:

```
BTM   COMPLT, 0, 10
```

Thus, the FORTRAN INSTRUCTION:

```
READ n, A, B, I, X
```

would result in the following coding:

```
BTM   RACD, FORMAT +6
BTM   SWC, A
BTM   SWC, B
BTM   SWC, -I
BTM   SWC, X
BTM   COMPLT, 0, 10
```

---

* This is not an I/A flag; it is used by the I/O routine to determine the size and operations to be performed on the data prior to output or storage.

```
##JOB 5

##SPS

*PUNCH RESEQUENCED SOURCE DECK
*OUTPUT CARD
*LIST TYPEWRITER
*TYPE SYMBOL TABLE


                                SYMBOL TABLE

 COUNT 02565      GO    02402      INPUT 02553      KON   02559      START 02414

00010*
00020*
00030*              RESEQUENCER PROGRAM
00040*
00050*
00060 GO      TF        COUNT,KON                     02402 26 02565 02559
00070 START   RNCD      INPUT-79                      02414 36 02474 00500
00080         TF        INPUT,COUNT                   02426 26 02553 02565
00090         WNCD      INPUT-79                      02438 38 02474 00400
00100         AM        COUNT,1,10,                   02450 11 02565 00001
00110         B         START                         02462 49 02414 00000
00120 INPUT   DS        80                            02553 00080
00130 KON     DC        6,1                           02559 00006 000001
00140 COUNT   DS        6                             02565 00006
00150*
00160*
00170*              PROGRAM WRITTEN BY D.N. LEESON
00180*
00190*              REFERENCES
00200*
00210*              LEESON,D.           UNPUBLISHED LECTURE NOTES
00220*
00230*              WEPMAN,R.           UNPUBLISHED LECTURE NOTES
00240*
00250         DEND      GO                            02402

END OF ASSEMBLY.
02566 CORE POSITIONS REQUIRED
00025 STATEMENTS PROCESSED
```


Electric Utility Programs Team

Newsletter #19 18 December 64

Norman, Oklahoma Meeting
November, 1964

These notes are provided us by Larry Redmond of Southern Services who attended the meeting when your secretary had to be two places at once.

With Frank Wells, Long Island Lighting Company, presiding, the floor was immediately opened for NOMINATIONS for chairman. Al Lipson, Virginia Electric and Power Company, nominated Frank and moved that nominations be closed. Thus, Frank found himself the unanimous choice. Frank announced that he would appoint the secretary.

A tabulation of the ballots sent out after the Washington meeting for modifying Glimm's Load Flow indicates 17 for, and 10 against. Frank stated that this was not enough to make a decision. So, the question will be left up to the individual.

The FUTURE OF THE ELECTRIC UTILITIES PROGRAM TEAM is of vital concern to all and generated much discussion. The 360 will undoubtedly play an important part in molding the future of this group. In fact, a stranger attending the various sessions of the Oklahoma meeting would surmize that the Model II 1620 with disk and the 360 were the only machine configurations available.

Frank Wells stated his belief that there will be a slow changeover to the 360, but we will hold together. Frank also stated that we will keep independent affiliation.

At this point Bob White, Los Angeles Light and Power, pointed out the need to establish contact between the utilities group and other groups. Bob feels that definite liaison should be established. Frank Wells agreed to contact other groups and report at the next meeting.

Frank Wells made a report on the progress being made by 360 USERS. The 360 group is made up of the following companies:

1. American Electric Power,
2. Cleveland Electric Illuminating,
3. Commonwealth Edison, Chicago,
4. Consolidated Edison, New York,
5. Consumers Power,
6. Detroit Edison,
7. Ohio Edison,
8. Wisconsin Electric Power, and
9. TVA.

The IBM representative to this group will be Arno Glimm.

The 360 group met in Chicago November 5 to establish a program for development. Their next meeting will be at the IEEE Winter meeting in New York.

The method of developing programs is for a committee to establish program specifications. After the specifications are agreed upon, one company will finalize the program.

The aim for the group is to establish memory banks utilizing data cells with two to three billion character capacity. Programs and system data will be stored in cells and will be available for program solution by dialing the 360 computer via a local 1070 computer.

A library will be maintained by IBM. Programs submitted by 360 users will be classified as class 3 programs.

Bob Steinhart of IBM said the 360 people intend to first convert present applications to 360, then go to the data bank concept. Bob also stated that IBM will not recognize this group as a bonafide user group, but will give program support. It seems that this is the desire of the 360 group.

Languages for the 360 systems include 5 forms of FORTRAN, 3 assembliers, and 1 new program language. Model 30 FORTRAN is similar to 1410 FORTRAN IV and Model 60 FORTRAN is similar to 7090 FORTRAN IV.

Con Edison has ordered 3 Model 30's, 1 Model 50, and 1 Model 62. The Model 50 is the work horse for the Model 62 computer. Model 1070 computers will form the pipe line from the office to the 360 computer.

To date, 20 programs have been assigned for development. These include load flow, stability, probability and reliability, production costs, circuit analysis, short circuit, steam properties, generalized heat balance, tower spotting, pipe stress analysis, and load forecasting.

After some altercation, it was decided that OUR NEXT MEETING would be with the Power Industry Computer Application Conference which will be held May 19, 20, and 21, 1965, at the Jack Tar Harrison Hotel, Clearwater, Florida. The PICA brochure indicates that questions about this conference should be directed to Mr. Lester Ulm, Jr. or Mr. J. R. Brice, Tampa Electric Company, PO Box 111, Tampa, Florida 33601.

Someone suggested having the Utility Team meeting a day before the PICA Conference next year. This would not only enable us to attend all sessions of the PICA Conference, but would also give us more time together to "hash-over" our Team meeting. Frank Wells said he would look into the PICA meeting program and report to us later.

Emil F. Buettner and Eldon McCollom of Oklahoma Gas and Electric, presented "Complete Evaluation of System Performance Attained Utilizing 1620-1401 Computer Team". This presentation shared Oklahoma Gas and Electric's experiences with LOW VOLTAGE UNDERGROUND NETWORK STUDIES.

The starting point in a load flow analysis is the determination of the load. A statistical load analysis is run each fall on the 1401 based on the customers KWH consumption during the month of July or August. The 1401 assigns each individual load to the appropriate element of the electrical system. In overhead systems, the loads are assigned to specific poles and in underground systems the loads are assigned to specific manholes, vaults or service boxes.

Arno Glimn's 1620 load flow program is used. Due to 1620 limitations a maximum of 125 buses may be studied.

Utilizing a disk pack enables after hours operation with no operator intervention. The program loads automatically, solves the base case, solves each change case in reference to the base case and loads all solutions on disk.

It was pointed out that since change cards are read via the card reader, a reader check could stall the operation.

The problem of output analysis is handled by two specially written programs.

1. Summary Program:
   This SPS program summarizes the load flow output by identifying the bus numbers, assigning equipment ratings, and eliminating duplication of the load flow program. After system data is loaded into storage, the program read the load flow output cards, one at a time, and punches a new card for each cable, transformer, and bus on the system.

2. Load Program:
   This FORTRAN program calculates loads, compares loads to ratings, and flags all overloads and low voltage conditions. Output from this analysis program is listed on a 407.

Some trouble has been experienced in system updating. An updating process has been established with the engineer responsible. Updating is maintained in the division offices and all information sent to the main office.

Next, Jene Louis, Long Island Lighting Company, reported on the progress of the 20K VERSION OF HEAT BALANCE I written by Bob White and Yosh Fujimura of the Department of Water and Power, City of Los Angeles. Jene reports that with the help of an IBM summer employee, this job is coming along nicely. Progress as of the Oklahoma meeting was a big red light, but with Bob White's help, Jene hopes to have something for us by the next meeting. You will find details of this program in Newsletter #16.

The "big" Heat Balance Program is now available as No. 1620-9.5.009.

Jene Louis, LILCO, presented a SURVEY OF 1620 FORCOM SYSTEMS, a copy of which is attached.

Also attached is a copy of Mr. V. A. Lippo's letter to Frank Wells describing the 1620 DISC CONVERSION MADE BY WEST PENN POWER.

Bob Steinhart reported that the 1401 PRINT PROGRAM FOR THE 1620 ELECTRIC LOAD FLOW SOLUTION written by John Evans and James Pitts of Southwestern Public Service Company is available, numbered 1401-9.4.002. Mr. J. C. Claughton of Southwestern Public Service Company reviewed the program specifications which you will find in Newsletter #16.

Mr. M. E. Byrne of Idaho Power Company told us of their load flow package. This package consists of the main load flow program plus several auxiliary programs as follows.

I. Listing With Names
   a. This uses the output cards from the original load flow and adds names that are in columns 61-68 of the original bus cards.
   b. The output of this spread to a 120 on the 407 printer.

II. Map Printout
   a. This is a two-pass program with a sort in between passes.
   b. It is printed on the same 407 board as the listing with names.

III. Voltage Estimates-This is two small programs, one for voltage output and one for voltage input.

IV. Change bus/line transformer in generator configuration in the condensed deck
   a. This changes the number of buses/lines, etc., in the compressed decks.
   b. The first card of each deck carries the information as to the number of buses/lines, etc.
   c. The second card has the addresses of bus I, line 1, generator 1, transformer 1, so one program listing can be used with any configuration.

The load flow solve program has been altered to include an automatic variable acceleration factor under console switch control. This has been tested with the Long Island Company case that was used to test Glimn's original program and Fujimura's program with the following results.

	Glimn		Fujimura		Swenson	
	Iter	Time	Iter	Time	Iter	Time
Base Case	40	0:37	68	2:00	57	0:39
Change Case	Non-Convergence		95	2:35	76	0:49

*Thru-put with 500 cpm read and 250 cpm punch.

(Ed. Comment: The times under the Glimn and Fujimura headings (are apparently abstracted from Newsletter #13 (27Aug63). These (times are based on 250/125 cpm I/O. Soon after these figures (were first published, Yosh Fujimura replaced the square root (routine in his program with an approximation, which greatly cut (down running time. For additional comments on these two sets (of figures, see Newsletter #13.

Mr. Byrne suggests that you check the tap changing routine in your solve program to make sure that the line record is returned to the line block the admittance of that change in the same direction. Mr. Byrne offers the following correction to the solve program to Glimn's load flow.

```
18360  26  12033  00093  26060  TF  B,93
                              TF  *+18, Line + 11
          Addition ------>    TR      , LINREC
18372  31  00000  12039  26070  TR      , TRNREC
```

Bob Steinhart says that he will help push programs into the library. Send him a copy of the letter of submission and he will do all he can to insure early availability.

Ed Cox, Memphis Light and Gas, asks if anyone has done anything with the Puerto Rico POSITIVE SEQUENCE SHORT CIRCUIT PROGRAM. It seems that it will not solve radial lines. After a moment of silence, Ed stated that he was looking into the problem, but thus far has not found the solution.

R. E. Smith of Gulf States Utilities gave a very interesting presentation of their COMPUTER CONTROL DISPATCHING SYSTEM.

In 1961, with the installation of an IBM 1620 computer, Gulf States initiated a long-range "pay as you go" plan. This step-wise plan not only allows each step to prove itself profitable, but also gives time to adjust procedures and to make optimum use of new equipment.

In January, 1965, with the installation of an IBM 1710 system, real time dispatching will be realized. In making a thorough analysis of the system to determine total load distribution this system considers transmission losses, fuel prices at each of the plants, gas contracts, unit incremental heat rates, and tie line flows.

Ultimately, Gulf States plans to install an IBM 360 System which will not only give them increased speed and accuracy in dispatching, but also allow more sophisticated off-line computing.

Studies prove this dispatching system will save more than one hundred thousand dollars per year in fuel costs alone.

This system will be described in detail in an article written for Electrical World by A. J. Mary and R. E. Smith of Gulf States Utilities and F. E. Hurlburt of IBM Corporation.

Larry Redmond described Southern Services TOWER SPOTTING progress. Southern Services has converted the Cleveland Electric Illuminating Company's program written for a 10K 7074 to the 16K IBM 7040. Survey data is punched directly from the survey notes and edited with a FORTRAN II 1620 program. A 7040 plot program is being developed to plot the profile complete with alphabetic survey comments as a strip chart to the same scale as that used for manual spotting.

In a test case, spotting 3 towers and keeping 1, a speed of approximately 25 mph was realized. This case covered a distance of 10 miles, spotted the same number of towers as a manual spotting, and gave shorter heights. Indicated savings over this very regular profile were about 3%.

Larry also reported on Southern Services progress using LINEAR PROGRAMMING TO GENERATE FUEL SCHEDULES. The IBM written program, Transportation Problems-Indirect Addressing No. 1620-LM-017 is being used. The output of this program has been "doctored" to give a more readable format to the fuel engineer.

Other changes to this program include the ability to consider many different alternates by using change cards with no operator intervention and the ability to solve step rates (accomplished by an iterative technique). To date, 51 sources and 8 steam plants is the largest case run. On Model II this solution takes about 15 seconds.

Anyone desiring more information on these two Southern Services applications may contact your secretary.

At this point, someone made a motion for the door and the formal Utility Programs Team Meeting came to an end.

Frank Wells indicates that an impromptu meeting was held Wednesday morning. In a letter Frank describes some bits of information from this get-together. "Dick Smith of Gulf States Utilities and Al Lipson of Virginia Electric Power debated the use of a stepped incremental heat rate curve versus a smoothed curve for economic dispatch. Apparently this is a fielder's choice, since the difference in results is insignificant. Another problem for discussion, presented by Al Lipson, involved the economics of coal distribution from mine to plant with multiple coal sources and coal prices and several plant locations. Transportation costs are alike to any plant. Al suggested that the highest price coal should be shipped to the highest efficiency plant to obtain the greatest kwhr output per pound of coal. Others stated that the price of coal was not a factor for the economic operation of the plants.

"Al also brought up the problem of calculating voltage gradients of bundled conductors for EHV lines. There are papers on such calculations as far back as 1922 which can serve as guides. The coupling effects of adjacent power lines, where voltages are shifted 30 degrees due to transformer connections, add to the complications.

"Eli Katz, City of Los Angeles, advised that he is considering a paper for the PICA meeting on the subject of power generation allocation in a network by power superposition method. This may be useful for exploratory planning purposes. Eli asked whether this paper would be worthwhile. The consensus favored presentation."

- -

Thus, the Oklahoma meeting. Other items follow.

On December 1, 2, and 3 a meeting of the MECHANICS WORKING UNIT of the Task Force on Computer Applications of the EEI was held in Baltimore. Topics discussed:

1. A workshop on the Baltimore Gas and Electric-Detroit Edison Company Steel Structures Design System for the IBM 7094,
2. Transmission Tower Spotting,
3. Sag and Tension and Other Problems,
4. Design of Towers and Other Structures,
5. A discussion of the various problem oriented languages for analysis of structural design problems,
6. Pipe Stress,
7. Topographical Problems, and
8. Dynamic Analysis in general with particular application to the problems of galloping conductors and stack vibrations.

This was the first meeting of the newly constituted Mechanics Working Unit of the EEI Task Force on Computer Applications. If any one idea could be singled out as the theme of this three day meeting, it was the concept of computers as a design tool. That is, the use of computers to refine a design rather than to get a single answer to a problem. An example from the structural design will suffice: In designing a steam power plant the structural design cannot be optimized until location of all components is determined. Yet when this stage is reached, the press of time allows little optimization.

Just before closing on the final day, a group of working teams was set up to study the problems of pipe stress, design of small structures including substation structures, transmission tower spotting, and topological design. The task of these groups is to stay abreast of the developments in the field and to attempt to formulate a design type approach to their problem. Most existing approaches merely analyze a given situation. Ideally the problem should be described to the machine in a very briefest of terms and the machine allowed to present an optimum design. In the field of pipe stress, for instance, the allowable end point reactions, the operating pressure and temperature, the maximum pressure drop and a description of the space available could be presented to the computer. The output would be a design which would fit within these restraints and give the minimum costs. The computer would consider such things as various types of steels, various routings, and configuration of the system.

Only one criterion was established for attendance: that a company be interested in the items set down for discussion. Therefore a lively and fruitful meeting resulted. The organizers of this group hope to be able to cut down on redundant efforts in their fields of activity. If anyone is interested, they

may contact Mr. D. D. Williams, Baltimore Gas and Electric Company, Constitution Street Building, Room 306, Baltimore, Maryland 21203.

Southern Services PIPE STRESS program will be available from the IBM library after 16 December: Number 1620-9.5.012, entitled "General Pipe Stress". This program is for 60K card in FORTRAN II. If you are interested in another configuration or another machine, please write your secretary. Several efforts are afoot.

Don Cahalan, Black and Veatch, P. O. Box 8405, Kansas City, Missouri 64114, is interested in a program for design of single circuit transmission towers. Will anyone having any leads please contact him?

Attached is a ballot for Chairman. You may either X Frank's name or write in your choice. Please return to your secretary by January 15, 1965. To mail, fold the bottom section back and then the top section back, staple, attach a stamp, and drop in the mail.

Sincerely,

Ed Orth

Ed Orth
Team Secretary
Southern Services, Inc.
PO Box 2641
Birmingham, Alabama

Attachments

TEAM MEMBERS AND GUESTS

UNIVERSITY OF OKLAHOMA MEETING
NOVEMBER 10, 11, 1964

1. Maurice E. Bryne, Idaho Power Co., 5115
2. Emil Buettner, Oklahoma Gas & Electric Co., 3234
3. Donald L. Cahalan, Black & Veatch, 3207
4. C.E. Chick, Gulf States Utilities, 1106
5. Stanley A. Clark, Public Service Co., New Hampshire, 1252
6. J.C. Claughton, Southwest Public Service Co., 5063
7. Ed Cox, Memphis Light, Gas & Water Division, 1243
8. Dave Davison, Public Service Co. of Oklahoma, 3083
9. Thomas H. Farrow, Tampa Electric Co., 1015
10. I.L. Gibbons, Arkansas Power & Light Co., 1457
11. B.E. Green, Arkansas Power & Light Co., 1457
12. George S. Haralampu, New England Electric System, 1041
13. Ralph Heyler, Southwest Public Service Co., 5063
14. Tom Hoke, Oklahoma Gas & Electric Co., 3234
15. W.M. Jones, Arkansas Power & Light Co., 1457
16. Eli Katz, Los Angeles Dept. of Water & Power, 5181
17. William D. Lang, Florida Power & Light, 1225
18. Alvin L. Lipson, Virginia Electric Power Co., 1044
19. Jene Y. Louis, Long Island Lighting Co., 1120
20. W.B. Martin, Oklahoma Gas & Electric Co., 3234
21. Eldon McCollum, Oklahoma Gas & Electric Co., 3234
22. Ben McMullin, Southwest Public Service Co., 5063
23. J.D. Pendergrass, Southwest Power Admin., 3118
24. Quo Philip, Black & Veatch, 3207
25. Morris Rantz, Southwest Public Service Co., 5063
26. John L. Redmond, Southern Services Inc., 1125
27. E.E. Riggin, Southwest Power Admin., 3118
28. R.E. Smith, Gulf States Utilities, 1106
29. R.F. Steinhart, I.B.M., N.Y.
30. F. J. Wells, Long Island Lighting Co., 1120
31. Robert R. White, Los Angeles Dept. of Water & Power, 5181

WEST PENN POWER COMPANY

CABIN HILL, GREENSBURG, PA. 15602

TEmple 7-3000 • Dial Code 412

October 23, 1964

Mr. Frank J. Wells
Long Island Lighting Company
175 Old Country Road, Hicksville
Long Island, New York

Dear Frank:

1620 DISK CONVERSION

West Penn has converted its 1620 40K system to 20K disk file system with no change in rental cost.

SPS and machine language programs over 20K are broken into 20K sections and instructions added to save intermediate answers and to bring into the core, from the disk, the proper sections of the data and/or programs at execution time. Running time losses up to 10-15% were experienced for 40K programs, but time savings up to 75% were made for programs which were run earlier in several passes.

For FORTRAN, special system called DEFT (Disk Execution of FORTRAN Transfers) was developed by West Penn. Programs were broken into sections of 20K which were processed independently, then stored in the disk and transferred when needed between core and disk in core image in modules of 20K at a time. The transfer package consists of machine language instructions which overlay the first 300 digits of every 20K section of FORTRAN programs. The time required to save data in the disk, to bring in new 20K program, and to return data to core is about .8 sec. without checking, 1.25 sec. with checking for programs of less than 10 sections. The program size is limited to 99 FORTRAN sections, 20K each.

The first use of the DEFT system was for a meter survey study, where in a 4 1/2 hour run, a 700 digit data block was transferred to disk, returned from disk, and checked both ways 6,000 times; and 20,000 digit program sections were brought from the disk and checked 6,000 times. The disk transfers took approximately 2 hours. These transfers would have taken 140 hours if done with multipass programming without the disk.

Mr. Williams advocated in the newsletter of October 14 breaking programs down to modules. It may be apparent that we have been doing just that in our system. Enclosed is a list of completed major programs, larger than 20K, which we have broken into sections of 20K, provided with proper disk loader and starter packages, and permanently stored in disk. Some of our programs are being automated so that if a code card is placed at the end of the data deck, a new program is read in from the disk as designated by the starter card(s) after the code card.

Yours truly,

V. A. Lippo, Supervisor
Engineering Computing

VAL: tmh
Enc.

LIST OF SECTIONALIZED OVER 20K PROGRAMS

	Language	Sections
Transmission Tower Design	SPS	3
Sag & Tension	ML	2
Economic Dispatch	SPS	7
Power Production Cost	ML	2
1-2 Feed Point Faults	F	2
3-Feed Point Faults	F	2
Short Circuit Calculation	SPS	7
Westinghouse Short Circuit Data Converter	SPS	4
Flood Spillage Report	F	2
Space Conditioning by Heat Pump	SPS	2
150/290 Load Flow	SPS	7
Daily Load Forecasting	SPS	2
Computer Monthly Report	SPS	2
PERT - package	ML	3
WR-4 Meter Survey	F	2

GRAPHICS DEVICES

IBM 1015 Inquiry Display Terminal, used with 1410, 7010, and System/360. (Models 30, 40, 50).
This is a terminal through which a data processing system may be interrogated and on whose CRT replies may be displayed visually. Only alphameric characters may be displayed. Up to 1160 characters may be displayed simultaneously, and these are transmitted at up to 650 characters per second.

IBM 1627 Plotter, used with 1620 and 1401.

IBM 2250 Display Unit, used with System/360.
This is a CRT, with optional light pen feature, for displaying alphameric and/or graphic output. Contains 12" x 12" display area which may contain up to 52 lines of 74 characters each or whose 1024 x 1024 addressable points may be used in any manner.

IBM 2280 Film Recorder, used with System/360.
This unit records graphic and alphameric output on microfilm. After recording, the microfilm may be developed by an internal film processor and viewed on a built-in rear projection screen within 48 seconds. The film processor, as well as the projection facility, may be by-passed when off-line film processing is desired.

IBM 2281 Film Scanner, used with System/360.
This unit scans images on microfilm, such as charts, drawings, graphs, etc., and, under program control, digitizes them for direct input into System/360 for analysis and processing.

IBM 2282 Film Recorder/Scanner, used with System/360.
Combines the functions of the 2280 and 2281.

IBM 7404 Graphic Output Unit
This is a plotter which may be used on-line with the 7040, 7044, 7090, 7094, or 7094 II and also may be driven by the IBM 729 (556 cpi) or IBM 2400 Magnetic Tape Unit.

IBM 1403 Printer, used with 1401, 1410, 1440, 1460, 7010, 7040, 7044, or System/360.
A special chain which makes possible the printing of both alphameric and graphic output can be obtained.

RELATED EQUIPMENT

Digital storage of graphic information such as maps, drawings, graphs, etc. requires relatively low-cost, large-capacity, random-access equipment. Normal disk and magnetic tape equipment, while capable of handling this application, have been augmented by a new unit, the IBM 2321 Data Cell Drive. This unit, which may be used with the 1410, 7010, and System/360, has the capacity to hold up to 800,000,000 numeric digits available to the computer at any one time. Each drive holds up to ten removable and interchangeable data cells, enabling the operator to make an unlimited quantity of information available to the computer, and in bigger segments than are feasible with disks or tapes.

SURVEY OF 1620 FORCOM SYSTEMS

By J. Y. Louis
Long Island Lighting Company

I am sure that all of you agree with me that writing a normal size program in Fortran is very much easier than writing it in SPS or machine language. Not only does a SPS or machine language program take much more time to write compared to the time required to write the same program in Fortran, but also the debugging time for a Fortran program is usually only a fraction of the time required to debug the same program which is written in SPS or machine language.

After the IBM 1620 Fortran without Format came out, most of us realized that it is a very good system, very easy to write and fast running. However, one big drawback is no input and output format control. Because of the input and output format control requirement coupled with the necessities of alphanumeric symbol manipulation and card image manipulation at the Fortran level, the first Forcom system was developed.

Mr. R. K. Louden of IBM Detroit North presented the original 1620 Forcom System in the September 1961 Midwestern Region Meeting of the 1620 Users Group in Chicago, Illinois. The word "FORCOM" stands for "FORtran COMmercial." It is a group of Fortran subroutines designed to satisfy three objectives:

1. To provide a general purpose interpreter for commercial data processing operations.

2. To provide a complete input-output editing package for scientific applications in the IBM Fortran without Format language.

3. To provide alphameric symbol manipulation facilities at the Fortran level for logical applications which previously required machine language programming.

This Forcom system was also described in Issue 57 of the IBM Midwestern Region 1620 Technical Publications on System Engineering. This publication was reprinted in the Proceedings of the October 1961 Meeting of the Eastern Region of the 1620 Users Group at Boston, Massachusetts. Mr. G. Magnuson of IBM Chicago again presented the similar information for Mr. R. K. Louden in the January 1962 Western Region 1620 Users Group Meeting at San Francisco, California.

The original Forcom system include 14 subroutines: RCD, PCH, RTC, TAB, SPC, GET, PUT, ZON, PAS, TYP, UNL, ENT, CMP and CDS. The operations of these subroutines are attached.

Mr. W. L. Pope of Utah State University, Logan, Utah prepared similar Forcom subroutines for IBM Fortran with Format in 1962. He added the ABS subroutine and modified the PUT subroutine to detect when leftmost (most significant) digits of a number have been dropped.

Mr. J. S. Webster, State Electricity Commission of Victoria, Melbourne, Australia extended the CDS subroutine for possible permanent self-clearing on card images. He also extended the use of core storage space between normal card images. He adapted the original 14 Forcom subroutines with his new development to the IBM Fortran II in 1963. With the extended CDS subroutine, he replaced the CDS subroutine in the original Forcom package of Mr. Louden and replaced the CDS subroutine in the Forcom package for IBM Fortran with Format by Mr. Pope this year.

Mr. B. Betz, United Aircraft Corp., Norwalk, Connecticut developed 17 call type subroutines for IBM Fortran II-D for the IBM 1620/1311 system. He called it "Forcom II-D subroutines for Fortran II-D-Card." See attached summary.

At present, to the best of my knowledge, PDQ Fortran is about the best 20K 1620 Format Fortran available and Auto Float Fortran is the fastest running non-format Fortran for a 20K 1620 computer with special hardwares. We have a 20K 1620 computer with special hardwares. Therefore, we adapted the Forcom system to both of these Fortran systems. These Forcom systems can handle all the 14 original Forcom subroutines with the following exceptions. Because of the structure of the PDQ and Auto Float Fortran Systems, we have to use A=B*1.0 instead of A=B when B is put into the temporary accumulator. Also the UNL subroutine in our Auto Float Forcom can only handle four characters rather than five characters as in the original Forcom.

Summary of Fortran Subroutines

I. For IBM Fortran without Format by Mr. R. K. Louden: This system includes 14 subroutines:

   (1) RCD, which reads 80 column cards into any of nine different card image areas with last card indication.

   (2) PCH, which punches cards from the above nine card image areas.

   (3) RTC, which returns the 1620 typewriter carriage.

   (4) TAB, which tabulates the typewriter.

   (5) SPC, which spaces the typewriter.

(6) GET, which gets numeric variable length fields from images and stores the fields as floating point variables.

(7) PUT, which puts floating point variables into card images as variable length numeric fields.

(8) ZON, which transmits zone punches to and from card images.

(9) PAS, which passes groups of characters between card images.

(10) TYP, which types any group of characters from any card image upon the 1620 typewriter.

(11) UNL, which unloads alphameric fields to and from card images.

(12) ENT, which enters data from the 1620 typewriter into the card images.

(13) CMP, which compares variable length alphameric fields in card images.

(14) CDS, which reserves card images in core storage.

II. For IBM Fortran with Format by Mr. W. L. Pope: This system has the same subroutines as those in Mr. Louden's original package with two exceptions:

(1) The absolute value (ABS) subroutine was added, and
(2) The PUT subroutine was modified to detect when leftmost (most significant) digits of number were dropped. This feature can be overrided by the users' choice.

III. For IBM Fortran II by Mr. J. S. Webster: This system was adapted from the original Fortran without Format by Mr. R. K. Louden with two exceptions:

(1) All subroutine names end with a letter F, such as CDSF and RCDF for CDS and RCD etc.

(2) The CDSF subroutine was modified for possible permanent self-clearing on card images and for possible use of core storage space between normal card images.

IV. For IBM Fortran without Format and IBM Fortran with Format by Mr. J. S. Webster: The CDS subroutine for IBM Fortran without Format by Mr. Louden and the CDS subroutine for IBM Fortran with Format by Mr. Pope were modified for possible permanent self-clearing on card images and for possible use of core storage apace between normal card images.

V. For PDQ Fortran by Long Island Lighting Company: This system has the same subroutines as those in Mr. Louden's original package except that we have to use A=B*1.0 in this system instead of A=B when B is put into the temporary accumulator.

VI. For Auto Float Fortran by Long Island Lighting Company: This system has the same subroutines as those in Mr. Pope's package with three exceptions:

(1) A=B*1.0 must be used instead of A=B when B is put into the temporary accumulator.

(2) The detectionfor digits of number being dropped in the PUT subroutine was deleted.

(3) UNL subroutine can only handle four characters.

VII. For IBM Fortran II-D by Mr. B. Betz: This system consists of 17 call type subroutines:

(1) CALL FORCOM, which causes 10 card areas (0-9) to be loaded with an object program.

(2) CALL READ, which reads cards into card image areas with last card indication.

(3) CALL PUNCH, which punches cards from card image areas.

(4) CALL PAPER, which controls the position of the paper in the typewriter.

(5) CALL PASS, which moves information from one card area to another.

(6) CALL TYPE, which types card area contents.

(7) CALL ENTER, which allows information to be entered into card areas from the typewriter.

(8) CALL COMP, which allows from 1 to 80 card columns to be compared to 1 to 80 other card columns.

(9) CALL UNLOAD, which saves card column information in Fortran variables.

(10) CALL LOAD, which restores card columns stored by UNLOAD to card area.

Summary of Fortran Subroutines (Continued)

(11) CALL FLAG, which tests for the presence of 11 zones in card columns and to insert 11 zones in card columns.

(12) CALL TEST, which tests any number of card columns for a particular group of characters.

(13) CALL GET, which converts a field to a Fortran variable.

(14) CALL PUT, which puts Fortran floating point variables into card images.

(15) CALL GETFIX, to "GET" fixed point numbers.

(16) CALL PUTFIX, to "PUT" fixed point numbers.

(17) CALL DUMP, which types Forcom card images at an Error Stop.

REPORT OF THE FORTRAN - SPS - ALGOL INFORMATION EXCHANGE

This is the third report, and contains the information which has been gathered since the second report. Previous reports have appeared in the Proceedings of the February, 1963 meeting in Chicago and the October, 1963 meeting in Pittsburgh.

Response to this Exchange has not been as good as was hoped. Many of these items of information have been obtained in a round about way, rather than directly from the author, so they may not be entirely correct.

All persons making modifications to any of these systems, or planning to make such modifications, or planning to write new systems, are requested to notify this Exchange. Any person wishing information on any such project may obtain it by writing to the Exchange.

The intent of this Exchange is to eliminate redundant effort. It will be effective only if all such information is sent to it. Information, or requests for information, should be sent to

Mr. Lanny Hoffman
Guggenheim Laboratories
Forrestal Research Center
Princeton, New Jersey

SPS

A modified version of AFIT SPS (1.1.023), which does not require TNE, TNS, and MF, has been prepared. It requires 40K cards, IA, and auto divide. For further information, contact

Mr. David Olson
Computing Center
Newark College of Engineering
323 High Street
Newark 2, N.J.

The NCE High-Speed SPS Assembler is an attempt to provide an assembler with the language capabilities of AFIT SPS and a large symbol table on a 20K machine. It is described in the Proceedings of the May 1964 meeting in Washington. For further information, contact

Mr. Kurt Germann
Newark College of Engineering
323 High Street
Newark 2, New Jersey

A load-and-go SPS with complete error checking at object time, called MSC Assembly System, is described in the Proceedings of the June 1964 meeting in Denver. For further information, contact

Mr. Glenn R. Ingram
Department of Mathematics
Montana State College
Bozeman, Montana

AFIT SPS (1.1.023) has been modified for use with a 1443 printer. Numerous other improvements have been added, so that it has most of the language capabilities of SPS II-D. For further information, contact

Mr. Richard L. Pratt
Data Corporation
7500 Old Xenia Pike
Dayton, Ohio 45432

Further modifications to AFIT SPS for printer are being made by

Mr. James Stansbury
Halcon International
2 Park Avenue
New York, New York 10016

A modified SPS processor has been developed which operates faster and has a smaller deck. It is called FLBSPS (Faster, Less Bulky SPS). It requires 40K. Certain additional features are included. A printer version is also available. For further information, contact

Mr. Kenneth W. Jones
Senior Programmer
Colorado Department of Highways
4201 E. Arkansas Avenue
Denver 22, Colorado

A number of changes have been made to SPS to make it more efficient and easier to use. For further information, contact

Dr. W. Fassler
Contraves AG
Schaffhauserstrasse 580
Zurich 11/52, Switzerland

The 1620-1710 SPS processor has been modified to increase the size of the symbol table, and enable the user to use macro-instructions before a TCD statement, by

Dr. G.S.D. King
European Research Associates
Recherches Chimiques
Physiques et Metallurgiques
95 Rue Gatti De Gamond
Bruxelles 18, Belgium

A set of modifications of SP-020 aimed at greater efficiency on a basic hardware configuration has been completed. This results in a considerable time saving, at a loss of some of the features of SP-020. For further information, contact

Mr. Robert C. Babione
3965 Westminster Place
St. Louis, Missouri 63108

A load-and-go SPS for paper tape is being developed by

Dr. Ronald C. Read
University College of the West Indies
Mathematics Department
Mona, St. Andrew, Jamaica, W.I.

The INC and OUTC subroutines (1.6.053) have been modified for use with several different versions of SPS, including Monitor, by

Jane Grundy
U.S. Steel Corporation Research Lab.
Monroeville, Penna.

FORTRAN

AFIT Fortran (1.1.010) has been modified for use with a 1443 printer by

Mr. Noel T. Smith
Indiana State College
Terre Haute, Indiana

A polynomial subroutine for Fortran with Format, for polynomials of degree up to 8, has been written by

Mr. Ph. Passau
2 Allee De Platanes
~~Leverwol~~, Belgium
Loverval

A number of changes have been made to the Fortran with Format system to improve its efficiency and ease of use. They require indirect addressing and automatic divide. For further information, contact

Dr. W. Fassler
Contraves AG
Schaffhauserstrasse 580
Zurich 11/52, Switzerland

AFIT Fortran (1.1.010) has been modified for use with 1311 disk by

Mr. Charles Mandlin
Computing Center
University of Oklahoma
Norman, Okla.

A version of PDQ Fortran for a 1443 printer is available from

Mr. James S. Taylor
Data Corporation
7500 Old Xenia Pike
Dayton, Ohio 45432

A version of PDQ Fortran to include a complete pre-compiler, as well as other improvements, is being developed. It is still in an early stage of development. It will require 20K, cards, IA, automatic divide, and TNF, TNS, and MF. There will probably be a version without TNF, TNS, and MF. For further information, contact

Mr. Richard L. Pratt
Data Corporation
7500 Old Xenia Pike
Dayton, Ohio 45432

A Fortran IV compiler has been written for either magnetic tape or cards. It requires 60K, floating point hardware, and the features required for Fortran II. The system is available for field testing. For further information, contact

Mr. James White, Supervisor
Computer Facility
Mayo Clinic
Rochester, Minnesota

A Modification of Forgo (2.0.008) to use floating point hardware has been written. For information, contact

Mr. Herbert Alcorn
Missouri School of Mines
Rolla, Missouri

A version of PDQ Fortran (2.0.031) for paper tape has been written. It also includes additional language features. A description of it will be found in the Proceedings of the May 1964 meeting in Washington. For further information contact

Mr. J. W. Trantum
MRD Division
General American Transportation Corporation
7501 North Natchez Avenue
Niles, Illinois

A paper tape version of PDQ Fortran, with modified and simplified input-output, called SEX Fortran, has been written. It is described in the Proceedings of the May 1964 meeting in Washington. For further information, contact

Mr. P. G. Boekhoff
MRD Division
General American Transportation Corp.
7501 North Natchez Avenue
Niles, Illinois

A number of minor modifications to Fortran II to improve its usefulness have been made at Georgetown University. A routine is planned to produce an "SPS-like" listing of a Fortran II object program. For further information, contact

Mr. Donald L. Wright
Systems Analyst
Computation Center
Georgetown University
Washington, D.C. 20007

Programs to punch the symbol table in the IBM, UTO, and PDQ Fortran systems have been written. They are described in the Proceedings of the May 1964 meeting in Washington. For further information, contact

Mr. Richard C. Irons
U.S. Naval School of Aviation Medicine
U.S. Naval Aviation Medical Center
Pensacola, Florida

A program to list all references to labels and statement numbers in a Fortran program is being developed by

Mr. Frank H. Maskiell
Pennsylvania Transformer Div.
McGraw Edison Company
Canonsburg, Penna.

A PLOT subroutine has been written for NCE Load and Go Fortran (2.0.029). The listing of this subroutine, as well as an explanation of how to add other subroutines, will be found in the Proceedings of the May 1964 meeting in Washington. For further information, contact

Mr. Hubbard Seward
Newark College of Engineering
323 High Street
Newark 2, N.J.

A precompiler and statement number changer for PDQ Fortran is being written by

Mr. John O'Loughlin
Fort Hays State College
Fort Hays, Kansas

ALGOL

The SIU Algol system has been modified for use on a 40K Model 1, without TNS, TNF, and MF, but with automatic floating point, by

Mr. A.J. Rose
IBM
Thomas J. Watson Research Center
P. O. Box 218
Yorktown Heights, New York 10598

OTHER

A monitor supervisor for a disk system is described in the Proceedings of the May 1964 meeting in Washington. For further information, contact

Mr. E.E. Newman
Civil Engineering Dept.
MIT
77 Massachusetts Ave.
Cambridge, Mass.

A list processing system, called IPL-V, has been written for a 20K 1620 with card input-output, indirect addressing, automatic divide, and TNF, TNS, and MF. It is described in the Proceedings of the June 1964 meeting in Denver. For further information, contact

Mr. Wendell T. Beyer
University of Oregon
Eugene, Oregon

DOODLE is a three-address interpretive programming system. It is described in the Proceedings of the May 1964 meeting in Washington. For further information, contact

Mr. M.V. Forina
General Electric Company
600 Main St.
Johnson City, N.Y.