PROCEEDINGS
OF THE MEETING
OF THE WESTERN REGION
OF COMMON

The International Hotel
Los Angeles, California
December 6, 7, and 8, 1965

Table of Contents

PROGRAM AGENDA

MONDAY, DECEMBER 6

8:00 Late Registration

General Session

9:00 Welcome
IBM Announcements
Sound Off - Bob White

10:00 Coffee

10:30 Sound Off (contd.)

11:30 Converting from the 1620 to System 360 - The 1620 Compatability Feature for Model 30 by H. Elmer and H. Weber (IBM)

12:00 Lunch

	Session A	Session B
1:15	Civil Engineering Panel Richard Wainer, Chairman James W. Hunter Frank Julian James Nugent George Taylor Robert Olson and others	SNOBOL 3, A List Processing Language for the IBM 1620 by David L. Wilson
2:15		Easy LP by D. J. Aigner
2:40		1620 SPS-II and SPS II-D Object Deck Modifier by Betty M. Earlougher
3:00	Coffee	Coffee
3:30	Civil Engineering Panel (contd.)	The Omnimetrics Operating System by Marvin Rubinstein
4:30		Cleartran by Francis W. Winn
5:00	New User Meeting Paul Bickford	

TUESDAY, DECEMBER 7

	Session A	Session B	Session C
8:30	Education Panel George R. Jaffray, Chairman Carl Feingold (arrived late) John A. Ferling Fernando Figueroa Charles B. Kinzek William G. Lane	PERT and CPM Panel (see abstracts) Gaylord Baker, Chairman M. Lopez Marvin Rubinstein Dave Stadlman	Advanced Monitor Tutorial - Bert Madsen A class in how MONITOR is constructed and how to make modifications
10:00	Coffee	Coffee	Coffee
10:30	Panel (contd.)	Panel (contd.)	Tutorial (contd.)
11:00	Automonitor for 1620 Machine Language by John Rettenmayer	Solution of a Problem in Heat Transfer by James C. Caslin, H. E. Fettis, and John W. Goresh	
11:20	Anthropology and Programming by Herman B. Weissman	Computer Design of High Velocity Duct System by Ralph Vandiver	
11:50	The 1620 for Simulation in a Bio-Medical Environment by I. R. Neilsen and James J. Horning	Some Applications of Cracovians by Marco T. Rincon B.	
12:00			Expanding the Capability of Plotter Software by Dean Lawrence
12:10		Use of a 1620 at a Solar Observatory by Robert L. Shutt	
12:35	Lunch	Lunch	Lunch
2:00	Tour to Jet Propulsion Laboratory	Tour	Tour

WEDNESDAY, DECEMBER 8

General Session - Eli Katz

8:30 PL1 Tutorial - John Harris

10:00 Coffee

10:30 1130 Computer by Paul Manikowski

11:00 Sound Off Response - Bob White

12:00 Luncheon
Speaker: Fred Gruenberger
Subject: The Future of the Free Standing Small Computer

	Session A	Session B
1:45	Tutorial - 1130-1800 Programming - Paul Manikowski	Tutorial - 360 Programming Systems - Jay Michtom
3:00	Coffee	Coffee
3:30	Tutorial (contd.)	Tutorial (contd.)

1965 COMMON CONFERENCE - REGISTRATION

AIGNER, D.J.	UNIVERSITY OF ILLINOIS	3342
ALLEN, ROBERT C.	UNIVERSITY OF VICTORIA	7044
AMMERMAN, THOMAS W.	PITTSBURGH PLATE GLASS CO.	5195
BACON, RICHARD P.	SAN DIEGO STATE COLLEGE	5142
BAUER, EDWARD G.	SAN DIEGO STATE COLLEGE	5142
BELDNER, RUDYARD	LA DEPT OF WATER AND POWER	5181
BERRY, J. DOUGLASS	GANNETT FLEMING CORDDRY/CARPENTER	1009
BICKFORD, PAUL A.	OKLAHOMA STATE UNIVERSITY	
BIRD, FRANK	SACRAMENTO PEAK OBSERVATORY	5053
BOLES, JAMES N.	UNIVERSITY OF CALIF.,BERKELEY	5076
BROWNE, PAUL	UNION RESEARCH CENTER	5077
BRUCE, J.W.	OFFC OF SURVEYOR/ROAD COMM,SAN DIEGO	
BRYANT, JACK K.	LA DEPT. OF COUNT ENGR.	5018
BURNS, BRUCE A.	US AIR FORCE, COLORADO SPRINGS	1029
BUSCHMAN, W.O.	CALIF. STATE POLYTECHNIC COLLEGE, SAN LUIS OBISPO	
CAINSKI, RAY	PUBLIC SERVICE CO. OF NEW MEXICO	
CARROLL, LANE L.	SAN DIEGO STATE COLLEGE	5142
CHAMBERLAIN, D.L.	OFFC OF SURVEYOR/ROAD COMM,SAN DIEGO	
CLARK, CHARLES L.	CALIF STATE COLLEGE, LA	5185
CORTOPASSI, ANDREW	US BUREAU OF RECLAMATION	5096
COTTON, BILL	METAL STRUCTURES CORP	5208
CRABTREE, S. JAMES	SUNDSTRAND AVIATION, ROCKFORD, ILL.	
CURIEL, ROBERT	INFORMATION SYSTEMS CO.	5174
DABE, RODNEY G.	CONSOER, TOWNSEND + ASSOC.	3334
DAHLEN, JAMES W.	US ARMY CORPS OF ENGINEERS	5186
DICKSON, THOMAS R.	ORANGE COAST COLLEGE	5212
DIEHR, PAULA	ITT FEDERAL LABS	5227
DOUGHERTY, DANIEL J.	HAWAIIAN SUGAR PLANTERS ASSOC.	5030
EARLOUGHER, BETTY M.	STANFORD ELECTRONICS LAB	5123
ELMER, HANS	IBM, ENDICOTT	
ELY, DEAN E.	FRANKLIN ELECTRIC CO.,INC.	1432
EMERSON, JOHN T	FRESNO STATE COLLEGE	5241
FAGG, PETER	IBM, POUGHKEEPSIE	
FERLING, JOHN A.	CLAREMONT MENS COLLEGE	5033
FETTIS, HENRY E.	AERONAUTICAL RES. LAB	3024
GUINN, JOHN R.	TEXAS COLLEGE OF ARTS/INDUSTRIES	5104
HALL, OMER D.	LA COUNTY FLOOD CONTROL DISTRICT	5141
HECKMAN, A. R.	QUINTON ENGINEERS, LTD.	5054
HOLT, JOHN	NORTH AMERICAN AVIATION, INC.	5149
HORTON, MURIEL MRS	JET PROPULSION LAB	5019
HUNTER, JAMES W.	LA COUNTY	5018
HUSSAIN, KHATEEB M.	CALIF STATE COLLEGE, FULLERTON	
HUTZLER, R. H.	CONSULTANT - LOS ANGELES	
ISAACMAN, DAVID	QUINTON ENGINEERS, LTD.	5054
JACKSON, LARRY D.	COLORADO STATE UNIVERSITY	
JAFFRAY, GEORGE	LOS ANGELES VALLEY COLLEGE	
JOHNSTON, T. S.	TEXAS TECHNOLOGICAL COLLEGE	5143
JULIAN, F. B.	OFFC OF SURVEYOR/ROAD COMM,SAN DIEGO	
KATZ, ELI	LA DEPT. OF WATER AND POWER	5181
KISHI, HIROKO	WESTERN DATA PROC CENTER/UCLA	5215

KLEIN, SEYMOUR	LA CITY TRAFFIC DEPT.	
LAHNERS, ELAINE L.	VETERANS ADMIN. HOSPITAL	3055
LANE, WILLIAM G.	CHICO STATE COLLEGE	5190
LAWRENCE, DEAN	MIDWEST RESEARCH INST.	3180
LINSDAY, THOMAS H.	VENTURA COLLEGE	5243
LITTRELL, ROBERT T.	CALIF. STATE COLLEGE, LONG BEACH	5198
MAGWIRE, CRAIG	UNIVERSITY OF NEVADA	5038
MANIKOWSKI, PAUL	IBM, LA	
MARTINEZ, JESS	OMNIMETRICS	5172
MATTHEWS, E.L.	IBM, SAN JOSE, CALIF.	
MAUDLIN, CHARLES E.	UNIVERSITY OF OKLAHOMA, NORMAN, OKLA.	
MCCOLLUM, PAUL A.	OKLAHOMA STATE UNIVERSITY	3158
MCFARLAND, ALBERT	SAN DIEGO STATE COLLEGE	5142
MCMILLAN, FRANK	FRESNO STATE COLLEGE	5241
MCMENAMIN, JOSEPH L.	GROSSMONT COLLEGE	5145
MILLIGAN, PERCY L.	SOUTHERN UNIVERSITY	1339
MOFFITT, ROBERT D.	NORTH PACIFIC DIV.-CORPS OF ENGRGS	5085
MYLIUS, WILLIAM G. JR.	THE RUST ENGINEERING CO.	1164
NEAL, KENNETH L.	US ARMY CORP OF ENGR	5248
NORRIS, BOYD C.	US BUREAU OF RECLAMATION	5096
OLSON, ROBERT	LA DEPT OF WATER AND POWER	5181
PAQUIN, NANCY	US PUBLIC HEALTH SERVICE	1118
POTTS, W.W.	OFFC OF SURVEYOR/ROAD COMM, SAN DIEGO	
PRESTON, SPENCER V.	SAN DIEGO STATE COLLEGE	5142
RANDALL, ROBERT F.	AUSTIN COLLEGE	5252
REDLACK, HERBERT C.	SACRAMENTO STATE COLLEGE	
REICH, CARL T.	MONTEREY PENINSULA COLLEGE	5152
RICHARDS, THOMAS C.	VENTURA COLLEGE	5243
RINCON, MARCO TULIO	UNIVERSIDAD DEL ZULIA, VENEZUELA	8027
ROCKWELL, BILL	SAN DIEGO STATE COLLEGE	5142
ROEDER, GEORGE L.	US AIR FORCE, COLORADO SPRINGS	1029
RUBINSTEIN, MARVIN	OMNIMETRICS	5172
SAMSON, STEPHEN L.	US ATOMIC ENERGY COMM	1258
SARKISIAN, HARRY	HUMBOLDT STATE COLLEGE	
SCHANDUA, EMIL J.	ST. EDWARDS UNIV.	5228
SCHRADER, LUANA MRS.	CALIFORNIA STATE COLLEGE, HAYWARD	5105
SHUTT, ROBERT L.	SACRAMENTO PEAK OBSERVATORY	5053
STEINHILBER, J. R.	ITT FEDERAL LABS	5227
TAYLOR, GEORGE I.	LA COUNTY FLOOD CONTROL DISTRICT	5141
TOWN, GEORGE G.	SEATTLE UNIVERSITY	5219
TUCK, MICHAEL R.	ARGONNE NATIONAL LAB	1273
UTLEY, BRIAN	IBM, SAN JOSE	
VANCE, GARRY	UNIVERSITY OF NEVADA	5038
VANDIVER, RALPH JR.	BENHAM-BLAIR AND AFFILIATES, OKLAHOMA CITY	
VAURS, SANDRA	SAN JOSE STATE COLLEGE	5121
WAINER, RICHARD	DEPT OF PUBLIC WORKS, LA	5009
WALKER, CHARLES S.	ARIZONA STATE UNIV.	5199
WEISSMAN, HERMAN B.	UNIVERSITY OF ILLINOIS	
WHITE, ROBERT R.	LA DEPT. OF WATER AND POWER	5181
WILSON, DAVID L.	UNIVERSITY OF WISCONSIN	
WILSON, G.W.	MASSMAN CONST. CO., KANSAS CITY, MO.	3378
WINDMEYER, WALTER C.	DOW CHEMICAL CO.	5155
WINN, FRANCIS W.	COMPUTER LANGUAGE RESEARCH, DALLAS	5148
WOODWORTH, JAMES A.	DOW CHEMICAL CO	5155

1965 COMMON CONFERENCE - REGISTRATION

AIGNER, D.J.	UNIVERSITY OF ILLINOIS	URBANA, ILLINOIS
ALLEN, ROBERT C.	UNIVERSITY OF VICTORIA	VICTORIA,B.C.,CANADA
AMMERMAN, THOMAS W.	PITTSBURGH PLATE GLASS CO.	CORPUS CHRISTI, TEXAS
BACON, RICHARD P.	SAN DIEGO STATE COLLEGE	SAN DIEGO, CALIF.
BAUER, EDWARD G.	SAN DIEGO STATE COLLEGE	SAN DIEGO, CALIF.
BELDNER, RUDYARD	LA DEPT. OF WATER AND POWER	LOS ANGELES, CALIF.
BERRY, J. DOUGLASS	GANNETT FLEMING CORDDRY/CARPENTER	HARRISBURG, PENNA.
BICKFORD, PAUL A.	OKLA. STATE U. TECHNICAL INSTITUTE	OKLAHOMA CITY, OKLA.
BIRD, FRANK	SACRAMENTO PEAK OBSERVATORY	SUNSPOT, NEW MEXICO
BOLES, JAMES N.	UNIVERSITY OF CALIFORNIA, BERKELEY	BERKELEY, CALIF.
BROWNE, PAUL	UNION OIL RESEARCH CENTER	BREA, CALIF.
BRUCE, J. W.	OFFC OF SURVEYOR/ROAD COMMISSION	SAN DIEGO, CALIF.
BRYANT, JACK K.	LA DEPT. OF COUNTY ENGR.	LOS ANGELES, CALIF.
BURNS, BRUCE A.	US AIR FORCE	COLORADO SPRINGS,COLO.
BUSCHMAN, W.O.	CALIF. STATE POLYTECHNIC COLLEGE	SAN LUIS OBISPO,CALIF.
CAINSKI, RAY	PUBLIC SERVICE CO. OF NEW MEXICO	NEW MEXICO
CARROLL, LANE L.	SAN DIEGO STATE COLLEGE	SAN DIEGO, CALIF.
CHAMBERLAIN, D.L.	OFFC OF SURVEYOR/ROAD COMMISSION	SAN DIEGO, CALIF.
CLARK, CHARLES L.	CALIF STATE COLLEGE AT LA	LOS ANGELES, CALIF.
CORTOPASSI, ANDREW	US BUREAU OF RECLAMATION	SACRAMENTO, CALIF.
COTTON, BILL	METAL STRUCTURES CORP.	GRAPEVINE, TEXAS
CRABTREE, S. JAMES	SUNDSTRAND AVIATION	ROCKFORD, ILLINOIS
CURIEL, ROBERT	INFORMATION SYSTEMS CO.	LOS ANGELES, CALIF.
DABE, RODNEY G.	CONSOER, TOWNSEND + ASSOC.	CHICAGO, ILLINOIS
DAHLEN, JAMES W.	US ARMY CORPS OF ENGINEERS	SEATTLE, WASHINGTON
DICKSON, THOMAS R.	ORANGE COAST COLLEGE	COSTA MESA, CALIF.
DIEHR, PAULA	ITT FEDERAL LABS	SAN FERNANDO, CALIF.
DOUGHERTY,DANIEL J.	HAWAIIAN SUGAR PLANTERS ASSOC.	HONOLULU,HAWAII
EARLOUGHER,BETTY M.	STANFORD ELECTRONICS LAB	STANFORD, CALIF.
ELMER, HANS	IBM	ENDICOTT, NEW YORK
ELY, DEAN E.	FRANKLIN ELECTRIC CO.,INC.	BLUFFTON, INDIANA
EMERSON, JOHN T.	FRESNO STATE COLLEGE	FRESNO, CALIF.
FAGG, PETER	IBM	POUGHKEEPSIE,NEW YORK
FERLING, JOHN A.	CLAREMONT MENS COLLEGE	CLAREMONT, CALIF.
FETTIS, HENRY E.	AERONAUTICAL RES. LAB	W-PATTERSON AFB, OHIO
GUINN, JOHN R.	TEXAS COLLEGE OF ARTS/INDUSTRIES	KINGSVILLE, TEXAS
HALL, OMER D.	LA COUNTY FLOOD CONTROL DISTRICT	LOS ANGELES, CALIF.
HECKMAN, A.R.	QUINTON ENGINEERS, LTD.	LOS ANGELES, CALIF.
HOLT, JOHN	NORTH AMERICAN AVIATION, INC.	LOS ANGELES, CALIF.
HORTON, MURIEL	JET PROPULSION LAB	PASADENA, CALIF.
HUNTER, JAMES W.	LA COUNTY ENGR.	LOS ANGELES, CALIF.
HUSSAIN, KHATEEB M.	CALIF STATE COLLEGE AT FULLERTON	FULLERTON, CALIF.
HUTZLER, R.H.	CONSULTANT - LOS ANGELES	LOS ANGELES, CALIF.
ISAACMAN, DAVID	QUINTON ENGINEERS, LTD.	LOS ANGELES, CALIF.
JACKSON, LARRY D.	COLORADO STATE UNIVERSITY	FORT COLLINS, COLORADO
JAFFRAY, GEORGE	LOS ANGELES VALLEY COLLEGE	VAN NUYS, CALIF.
JOHNSTON, T.S.	TEXAS TECHNOLOGICAL COLLEGE	LUBBOCK, TEXAS
JULIAN, F.B.	OFFC OF SURVEYOR/ROAD COMMISSION	SAN DIEGO, CALIF.
KATZ, ELI	LA DEPT. OF WATER AND POWER	LOS ANGELES, CALIF.
KISHI, HIROKO	WESTERN DATA PROC CENTER/UCLA	LOS ANGELES, CALIF.

KLEIN, SEYMOUR	LA CITY TRAFFIC DEPT.	LOS ANGELES, CALIF.
LAHNERS, ELAINE L.	VETERANS ADMIN. HOSPITAL	OMAHA, NEBRASKA
LANE, WILLIAM G.	CHICO STATE COLLEGE	CHICO, CALIF.
LAWRENCE, DEAN	MIDWEST RESEARCH INST.	KANSAS CITY, MISSOURI
LINSDAY, THOMAS H.	VENTURA COLLEGE	VENTURA, CALIF.
LITTRELL, ROBERT T.	CALIF STATE COLLEGE AT LONG BEACH	LONG BEACH, CALIF.
MAGWIRE, CRAIG	UNIVERSITY OF NEVADA	RENO, NEVADA
MANIKOWSKI, PAUL	IBM WESTERN REGION	LOS ANGELES, CALIF.
MARTINEZ, JESS	OMNIMETRICS	LOS ANGELES, CALIF.
MATTHEWS, E.L.	IBM	SAN JOSE, CALIF.
MAUDLIN, CHARLES E.	UNIVERSITY OF OKLAHOMA	NORMAN, OKLAHOMA
MCCOLLUM, PAUL A.	OKLAHOMA STATE UNIVERSITY	STILLWATER, OKLAHOMA
MCFARLAND, ALBERT	SAN DIEGO STATE COLLEGE	SAN DIEGO, CALIF.
MCMILLAN, FRANK	FRESNO STATE COLLEGE	FRESNO, CALIF.
MCMENAMIN, JOSEPH L.	GROSSMONT COLLEGE	EL CAJON, CALIFORNIA
MILLIGAN, PERCY L.	SOUTHERN UNIVERSITY	BATON ROUGE, LOUISIANA
MOFFITT, ROBERT D.	NORTH PACIFIC DIV-CORPS OF ENGRS	PORTLAND, OREGON
MYLIUS, WILLIAM G.	THE RUST ENGINEERING CO.	BIRMINGHAM, ALABAMA
NEAL, KENNETH L.	US ARMY CORPS OF ENGR	SEATTLE, WASHINGTON
NORRIS, BOYD C.	US BUREAU OF RECLAMATION	SACRAMENTO, CALIF.
OLSON, ROBERT	LA DEPT OF WATER AND POWER	LOS ANGELES, CALIF.
PAQUIN, NANCY	US PUBLIC HEALTH SERVICE	ROCKVILLE, MARYLAND
POTTS, W.W.	OFFC OF SURVEYOR/ROAD COMMISSION	SAN DIEGO, CALIF.
PRESTON, SPENCER V.	SAN DIEGO STATE COLLEGE	SAN DIEGO, CALIF.
RANDALL, ROBERT F.	AUSTIN COLLEGE	SHERMAN, TEXAS
REDLACK, HERBERT C.	SACRAMENTO STATE COLLEGE	SACRAMENTO, CALIF.
REICH, CARL T.	MONTEREY PENINSULA COLLEGE	MONTEREY, CALIF.
RICHARDS, THOMAS C.	VENTURA COLLEGE	VENTURA, CALIF.
RINCON, MARCO TULIO	UNIVERSIDAD DEL ZULIA	MARACAIBO, VENEZUELA
ROCKWELL, BILL	SAN DIEGO STATE COLLEGE	SAN DIEGO, CALIF.
ROEDER, GEORGE L.	US AIR FORCE	COLORADO SPRINGS, COLO.
RUBINSTEIN, MARVIN	OMNIMETRICS	LOS ANGELES, CALIF.
SAMSON, STEPHEN L.	US ATOMIC ENERGY COMM.	NEW YORK, NEW YORK
SARKISIAN, HARRY	HUMBOLDT STATE COLLEGE	ARCATA, CALIF.
SCHANDUA, EMIL J.	ST. EDWARDS UNIVERSITY	AUSTIN, TEXAS
SCHRADER, LUANA	CALIF STATE COLLEGE AT HAYWARD	HAYWARD, CALIF.
SHUTT, ROBERT L.	SACRAMENTO PEAK OBSERVATORY	SUNSPOT, NEW MEXICO
STEINHILBER, J.R.	ITT FEDERAL LABS	SAN FERNANDO, CALIF.
TAYLOR, GEORGE I.	LA COUNTY FLOOD CONTROL DISTRICT	LOS ANGELES, CALIF.
TOWN, GEORGE G.	SEATTLE UNIVERSITY	SEATTLE, CALIF.
TUCK, MICHAEL R.	ARGONNE NATIONAL LAB	IDAHO FALLS, IDAHO
UTLEY, BRIAN	IBM	SAN JOSE, CALIF.
VANCE, GARRY	UNIVERSITY OF NEVADA	RENO, NEVADA
VANDIVER, RALPH JR.	BENHAM-BLAIR AND AFFILIATES	OKLAHOMA CITY, OKLA.
VAURS, SANDRA	SAN JOSE STATE COLLEGE	SAN JOSE, CALIF.
WAINER, RICHARD	DEPT OF PUBLIC WORKS	LOS ANGELES, CALIF.
WALKER, CHARLES S.	ARIZONA STATE UNIVERSITY	TEMPE, ARIZONA
WEISSMAN, HERMAN B.	UNIVERSITY OF ILLINOIS	URBANA, ILLINOIS
WHITE, ROBERT R.	LA DEPT OF WATER AND POWER	LOS ANGELES, CALIF.
WILSON, DAVID L.	UNIVERSITY OF WISCONSIN	MILWAUKEE, WISCONSIN
WILSON, G.W.	MASSMAN CONST. CO.	KANSAS CITY, MISSOURI
WINDMEYER, WALTER C.	DOW CHEMICAL CO.	HOUSTON, TEXAS
WINN, FRANCIS W.	COMPUTER LANGUAGE RESEARCH	DALLAS, TEXAS
WOODWORTH, JAMES A.	DOW CHEMICAL DO.	HOUSTON, TEXAS

Introduction to Civil Engineering Panel by Richard Wainer, Assistant Division Engineer, Coordinating Division, Bureau of Engineering of the City of Los Angeles.

The speakers on this panel, with the exception of Mr. Julian of San Diego, are from the Data Processing Committee of the Los Angeles Chapter of the American Public Works Association. About four years ago when the various governmental agencies in the area started to become computer oriented, informal meetings were held. To give a more official stature to the group, we requested recognition from the Los Angeles Chapter of the American Public Works Association and for the last three years we have been recognized as a working committee of this organization.

At the outset, workshop meetings were fairly frequent to share our experience and to prevent duplication of effort. As programs have been completed and familiarity with their usage obtained, the need for frequency of meetings has diminished and now meetings are held quarterly.

The Panel today will present information on some of the applications developing and problems encountered in the area of acceptance of computer technology in Municipal Engineering.

Mr. William Reader, Structural Engineer, Bridge and Structural Division, Bureau of Engineering, City of Los Angeles.

I want to discuss a method for the design of reinforced concrete multiple box structures. The slope deflection method used to develop the general algebraic equations is applicable to wide variety of structures though the loading conditions used are those found in conditions approximating underground and unsymetrical loading. Through the use of tabular design methods and computer

developed tables, the coefficients of maximum stress conditions are determined.

Fixed end moments are applied to each member and composite moment diagram is developed. Superpositioned moments are applied to balance the fixed end moments. The algebraic slope deflection equations are then reduced.

From these simplified algebraic equations, the computer (1620 Model 1, 20 K) was used to develop tables of coefficients to simplify the designer's work. The advantages of this method are:

1. Time savings and reduction of errors over previous methods.

2. An increase in visualization of effect of changes on final configuration.

3. Facilitate the development of more economical design.

Jim Nugent, Technical Service Supervisor Advance Planning Section, County of Los Angeles Road Department.

The County Road Department has developed and refined the classic problem of earth work into a neat package of both tabular output and graphic display using an 8K 1401.

This program reduces to a minimum the computational demands of the Engineer's time and frees him to evaluate the results and attempt better solutions.

The steps in establishing the economics of an earthwork design by the manual method are traditional: Alignment, both vertical and horizantal, is established by the Engineer; the cross sections are plotted and the template is superimposed; quantities are calculated from planimetered cross sections; and finally, the results are analyzed to determine the economics of the assumed

alignments. Often at this point, an entire reprocessing must be done to arrive at a better solution. Average Engineers' time for this manual processing is in the range of 65 hours per mile of roadway.

Through the application of new techniques, the cross section data is prepared photogrametrically, a trial alignment is made, and the information processed by our 1401 system. Cross sections and profiles are plotted by the 1403 at the rate of about 12 sections per minute, considerably faster than most line plotters. When the cross sections were plotted by a service bureau, the cost was about two dollars per section as compared to our current estimate of above ten cents a section.

This rapid and economic method allows for the establishment of the most economic design. It is now feasible to try several alignments or further refine a selected alignment to produce the minimum construction cost design. The computer time required for the automated method is about 20 minutes per mile including plotting.

Jim Hunder: Associate Civil Engineer, Office of the County Engineer, County of Los Angeles.

One of the typical problems of municipal Sanitation Design is the investigation of a sewer network. The flows, capacity, hydraulic grades and total flow at an outlet for a network are analyzed by the computer freeing the engineer of the relatively routine arithmetrical computations required.

The program developes a model of the system based on conduit size, land use, manhole numbers, drainage areas and slopes. This data, combined with actual measured data at control points,

produces a tabular output of percent of capacity being utilized. A tabular output is also produced showing what reaches have exceeded capacity and are in need of replacement.

As the program now operates on our 20K system, it is a three pass processing program. The first program accepts the manhole numbers, land use, pipe parameters and drainage areas and computes the theoretical capacity of the network elements. Pass 2 accepts the pass 1 output plus measured data to develop the percent of capacity being utilized. Pass 3 developes the formal output, replacement schedules, and estimated cost.

It is estimated that the cost saving realized over former methods is about 50 percent. In addition, once the basic data file has been established it becomes economically feasible to provide estimates of how zoning changes or new connections would affect the existing system. This method could be amplified to cover other gravity flow systems such as storm drains and water supply.

Frank B. Julian, Program Developement Engineer, Office of the County Road Commissioner, County of San Diego.

When I was requested to appear on this panel, my first idea was to speak about our 1620 earthwork program. After further thought, I changed my mind, but not because of a conflict with Mr. Nugent's presentation. The decision was based on a certain aspect of our program that presents problems familiar to most organizations. The documentation for input format is good, but due to lack of capacity, certain data checks are not made. When these conditions arise, naturally the program blows and the only one that knows where and how to correct it is the programmer himself.

Though the action of the computer can be reasonably anticipated, the human reaction is completely unpredictable. When we first installed the 1620 computer about four years ago, we had the green eye-shade and sleeve garter types that were still not too sure of the ability of a slide rule. As a consequence, no doors were broken down by the crushing hoard wanting to use the beast. Interest has been stimulated through familiarization classes and allowing them to write programs. The biggest selling point, though, has been short turn around time on data submitted.

At one end of the scale are those, and most are guilty, who enjoy writing a program, no matter how trivial, without a real understanding or interest in how much it costs. A singular example of this is the processor manipulators and program modifiers striving to save micro-seconds. This is fun but not really productive.

At the opposite extreme is the man who is afraid that the computer, like an expendible item of supplies, will be used up. He keeps looking for the great applications. Trivial applications, though productive and economically justified, are not of sufficient challenge to warrant implimentation. He over studies applications hoping the requestor will give up and go away.

As we discuss the various odd computer types, the last one to touch on is the non-documenting programmer who, four years later can not figure out his criptic notes on the edge of a sheet of paper. Use the ability of the processor to handle comment cards, the next man who has to pick up the program will be eternally grateful.

I am sure that these types are not unique to our organization, every installation has its share. The only thing we can all try to do is to place the computer in its proper prospective and pay more attention to the people who make up the organization.

Robert Olson, Water Works Engineer, Los Angeles Department of Water and Power.

I want to discuss a program for the flow distribution and head losses in a water distribution system on a 1620-60K system. Capacity of the program is 600 lines, 600 junctions and 175 loops. The solution is by means of the Hardy-Cross method of successive itterations. Input data is pipe parameters, roughness factors, inflows, outflows, and elevations of junctions.

The method of solution is to prepare a schematic of the network. The loops are numbered in an arbitrary fashion but not skipping any numbers. Starting with loop 1 a direction of flow is assumed for each line. Then each line is numbered. Each junction is numbered with junction #1 being a point of known hydraulic grade. On each line place the length, diameter and friction factor. Quantities of inflow and take-outs are shown and a check made to be sure that outflows are the same as inflows. The program uses nine different types of data cards for such information as identification, pipe numbers, parameters, number of itterations, junction numbers, and loop definitions.

Output is in three parts. Part 1 is an optional output showing unbalanced heads at junctions after each itteration. Second output is line data showing flow and head less in feet per 1000 feet. Output 3 is the hydraulic grade line elevations and pressure head at each junction.

George Taylor Jr., Associate Civil Engineer, Data Processing Section, Los Angeles County Flood Control District.

The Los Angeles County Flood Control District has modified the IBM-COGO-1 System from the original typewriter output to card output for offline listing because our workload required several hours of typing each day. Our workload has been running about 150% utilization.

Mr. Ramsdale of IBM and Mr. Lebow of the Los Angeles County Road Department have assisted in a further modification to use the 1443 Printer directly as the output device, by direct machine language patches which change the output device code from 04 to 09 and control the carriage skipping.

Had the COGO I System been compiled in FORTRAN with Format, it would have been possible to change the subroutine linkage address from type to Punch, since both statements compile as follows:

BT to the type or punch routine with the Format address,

BTM to the fill routine with each variable in turn except the last followed, by a BTM to the completion routine with the last variable.

The BT to the output routine could be changed by changing the linkage address every time it occured in a program. A simple SPS program was used to do this to one of our condensed program decks.

Fortran without Format compiles a type statement as follows: BT to a RCTY routine with the first variable, BT to a TBTY routine for each variable. The Punch statement, however,

compiles as follows:

BT to a fill routine for each variable, followed by a BT to the output routine.

This necessitated completely recompiling the COGO I System, changing all of the Type statements to Punch statements. A couple of the plug decks require executing them before punching the plug. This was done by manually branching directly to the proper statement number, the location of which was found by searching the symbol table.

A further modification is being made in the COGO I Monitor for checking the sequence numbers of the plugs decks as they are read (our 1622 Automatically shuffles the decks as they are read), automatically skipping to a new sheet when a character appears in column 80 of a data card, and to pack more than four variables into the output area similar to MIT's DTM system to simulate the original typewriter output format.

SNOBOL 3, A List Processing Language
For the IBM 1620

I. General Comparison--List Processing vs. Fortran.

A list processing language is designed to work with ALPHABETIC rather than numeric data as is FORTRAN. One would never consider writing a multiple correlation in SNOBOL. On the other hand, one could write programs in SNOBOL to:

1. Columnize an article for a newspaper. That is, set up the print so that the left and right margins are even and, if necessary, decide where to hyphenate words.

2. Read in a Fortran-like expression, take its analytic derivative, and put out a Fortran-like expression, which, when evaluated, will give the derivative.

Both of the above applications would be very difficult to programin FORTRAN or SPS.

II. Acknowledgement.

SNOBOL was first implemented for the IBM 7090 by D.J. Farker, R.E. Griswold, and I.P. Polonsky at Bell Labs, Holndel, New Jersey. There was an article written by them on SNOBOL 1 in the January 1964 issue of the Journal of the Association for Computing Machinery called "SNOBOL, A String Manipulation Language."

III. Direction of Implementation.

The purpose of the implementation is educational. We want to be able to tell our students what a list processing language is all about. As a result, this implementation is rather slow. Practical programs for production runs generally involve a change of card format involving about four SNOBOL statements.

For example, recently we ran a job with a group of address cards which had the last name first. The problem was to switch the last names down to the end. This involved a four card program in SNOBOL.

The program has been submitted to the 1620 Users Group Library. They have not yet finished processing it, but they have given it a temporary P.I.D. number of D3182.

IV. Data Structure--String and String Name.

The basic data structure of SNOBOL is the string. A string contains from 0 to 5000 ordered alphabetic, numeric, blank, or special characters. Each string has a name which is from 1 to 80 alphabetic or numeric characters including the special characters of period and record mark. The string is needed so that the SNOBOL program can refer to the string, much like a variable name in FORTRAN.

V. SNOBOL Statement Construction.

The SNOBOL statement consists of five parts:

LABEL	STRING	@ @	=	@≠@	/S(LABEL)
Statement Label	Reference String	Match Specification		Construction Specification	Go to Specification

Except for the End Statement, SNOBOL has just one statement type. The purpose of the separate parts of the statement is as follows:

1. Statement Label. A statement label must meet the same requirements as a string name, except that it must begin with a letter or digit. The statement label is needed for reference from other statements very much like statement numbers in FORTRAN. Like FORTRAN, where the statement number is optional, so is the statement label in SNOBOL. Statement labels must begin in col. 1 if present, otherwise col. 1 is left blank. The rest of the statement is in free format in the sense that wherever a space is required, more than one can be used.

2. Reference String Name. This gives the string which will be worked on for this statement. This is the only part of the statement which cannot be eliminated.

3. Match Specification: This specifies that which some substring in the reference string must match. In this case, the match specification has just one element: @ @. This is a literal string, which is specified by its contents surrounded by @ signs. Thus, in this case, we are looking for a blank somewhere in the string named STRING. The match scan is generally left to right, and the interpreter will take the first match it comes to. Since strings which contain @ signs cannot be specified by a literal, a special string QUOTE is supplied which contains just one @ sign. Notice that if there are no blanks in the string named STRING, the match "fails."

4. Construction Specification: This specifies what the substring which is matched is to be changed to in the reference string. This part of the statement is separated from the match specification by an equal sign. If the match specification is not included, the whole reference string is changed to what is specified. If the construction specification is omitted, then the reference string is left unchanged. If the match was unsuccessful, then the construction specification is ignored and the reference string is left unchanged. In this case, if a space is found in the string named STRING, it will be replaced by a record mark specified by @≠@.

5. Go to Specification. This specifies the next statement to be executed. If the part is omitted, the next sequential instruction in the program is executed. The slash preceded by a blank and followed by a non-blank character identifies the go to specification. Parentheses enclose the name of the next statement to be executed. This name can be constructed if desired in

/S(LABEL)

the S indicates a "conditional go to": go to LABEL if and only if the match was successful. A branch on failure is indicated by an F. Here the failure branch is not specified. Therefore, the next sequential instruction will be executed on a failure.

Since LABEL is the statement label of this statement, the interpreter will loop through the statement until all blanks are replaced by record marks in the string named STRING.

Notice that if the statement is modified to

LABEL STRING @ @ = /S(LABEL)

all the spaces in string will be deleted. This is one of the first things done in any FORTRAN compiler.

VI. Radix Sort Explanation.

The sample program (see inserted listing) which I am going to go through involves alphabetizing words using the radix sort technique. The technique is simply that used when one orders cards on a card sorter. For example, say one wanted to sort the following number into numeric order:

2297
2147
2293
2143
2197
2193
2243
2247.

One would first sort on the last column. (In case of ties original order is preserved):

2293
2143
2193
2243
2297
2147
2197
2247.

Then one would sort on the second last column:

2143
2243
2147
2247
2293
2297
2197.

Now the numbers are once again out of order with respect to the last digit. However, on the cards where the second last digit is the same, the cards are in order by the last digit, which is just what we want. Sorting on the third last digit gets:

2143
2147
2193
2197
2243
2247
2293
2297.

Since the first digits are all the same, an extra sort on the first digit will yield the same result. Notice that the numbers are now in order.

VII. Sample Program (see inserted listing).

I will go through the sample program on a card by card basis.

≠≠XEQ SNOBOL

This card would appear only on those machines which operate under a MONITOR system. Otherwise, it should be omitted.

BEGIN SYSPIT *SIZE* @ @

SYSPIT is a special string for the SYStem's Peripheral Input Tape. On this interpreter this means the card reader. Whenever SYSPIT appears, one card is read and its contents are put into a string of length 80. There is no construction or go to specification in this statement.

SIZE is what is known as a filler. It can be matched with as many characters from the reference string as necessary to permit the rest of the match specification to match. If the match is successful, then the contents of SIZE are replaced by those characters against which the filler was matched. A filler may match a null string (string of length zero). If a filler is the first element in a match specification list, then, as is this one, it is matched starting with the first character of the reference string; if it is the last element, the character it is matched against is extended to the last character of the reference string.

In this case, *SIZE* @ @ is being matched against the first input card which is a 9 followed by blanks. The @ @ will match the blank in col. 2 and SIZE will be filled with the 9. The

purpose of the statement is to read in the maximum number of letters in the word to be sorted or, looking at it another way, the number of columns to be sorted on.

```
START SYSPIT  *WORDS* @ @      /F(LO)
```

This statement is very much like the preceding one, only the names have been changed and a failure exit has been added. In this case, WORDS will be filled with ARMY, TEST, GLOBAL, etc.

```
LIST  =  LIST WORDS / (START)
```

Since the match specification is missing, the entire string is set equal to the construction specifications. Here the construction specification consists of two elements LIST and WORDS. The contents of the two strings are combined with all of the characters in LIST preceding all of the character in WORDS. The results are stored in LIST. Notice that the first time through the string, LIST is used before it has been defined. This perfectly legal. Any undefined string is taken as being null.

Then the program goes back to START and reads the next card whose contents, up to the first blank, are stored in WORDS. The contents of WORDS is then added onto the contents of LIST. Once again we go back to start. However, this time the card read doesn't have a blank. The contents of WORDS is left unchanged and control transfers to LO.

```
LO SYSPOT  =  @ THE LIST TO BE ALPHABETIZED IS - @ LIST
```

SYSPOT is a special symbol for SYStem's Peripheral Output Tape. In the case it's the typewriter, or, if you have a printer, the 1443 printer. There is also SYSPPT (SYStem's Peripheral Punch Tape), which is the card punch. Every time SYSPOT (or SYSPPT) appears in a statement, its contents are outputted once the statement completed. This statement will type out

```
THE LIST TO BE ALPHABETIZED IS-
```

followed by the contents of LIST.

```
SYSPOT  =  VOID
```

This prints a blank line since void is empty, being undefined.

```
L1  ALPHABET  =  @ABCDEFGHIJKLMNOPQRSTUVWXYZ@
```

This statement put the alphabet into ALPHABET. In the context of the program, the statement actually defines the collating sequence to be used. The definition is completely independent of the natural machine collating sequence.

```
L2  SIZE  =  SIZE  -  @1@
```

Arithmetic is permitted between two strings whose content are purely numeric. Both strings are decoded into ten-digit numbers; the arithmetic operation (+,-,*,/,or **) is carried out and the results are recoded as a string. Negative numbers are coded with a minus in front, all preceding zeros are dropped, except if the result is zero the resulting string is @0@. Special provision has been made to avoid coding a negative zero. Here SIZE, which was 9, is decremented to 8. SIZE contains the number of letters in a word to be ignored before the column we are now sorting on.

```
SIZE    @-@          /S(FIN)
```

This statement asks if there is a minus sign in SIZE. Eventually SIZE will be decremented to the point where it contains a minus one, in which case this statement will transfer to FIN. For the time being, the match will fail and we'll go on.

```
L3   LIST *WORD* @,@        =        /F(L5)
```

Everything up to the first comma in LIST is put into WORD and the substring of LIST which was matched, including the comma, will be set equal to the null string. That is, this particular work will be deleted from LIST. IF LIST HAS NO MORE IN IT, then the match fails and the program transfers to L5.

```
WORD *HEAD/SIZE*      *PIT/@1@*      /F(L4)
```

HEAD/SIZE denotes a "fixed length filler." In this case, HEAD can only be filled with the number of characters specified by the numeric string SIZE. Likewise, PIT can only be filled by one character, which happens to be the character we are sorting on. If the word is too small, the contents of HEAD and PIT are left unchanged and the program transfers to L4.

```
$ PIT   = $PIT WORD @,@ / (L3)
```

The $indicates indirect addressing. The contents of PIT are to be used as a string name. If PIT contains @K@, then it is the string name K that we are dealing with. Almost all restrictions on string names and statement labels are removed when indirect addressing is used. In this case, say we are sorting on the fourth letter of the work TENSOR. Then @TENSOR,@ would be added onto the string name S. These strings with one character names are being viewed as pockets in a sorter are. After this statement is done, it transfers back to L3 to pick off the next word from LIST.

```
L4   BIN    =    BIN WORD  @,@          /(L3)
```

BIN is used to store those words which are too short to be sorted at this time. Control is then transferred back to L3 to take the next word off of LIST.

```
L5   BIN *LIST* =
```

Now that all the words have been distributed among the "pockets," that is strings with one character names, we must collect the words back into LIST in collating sequence. This statement does two things:

1. Moves what is in BIN, that is, those words too short for sorting on this column, into LIST.

2. Set BIN equal to the null string.

This is done since the arbitrary filler LIST will match with the entire string BIN because it is both the first and last element in the match specification.

```
L6    ALPHABET  *PIT/@1@*  =  /F(11)
```

This statement put into PIT the next "pocket" to be put back into LIST. This letter is then deleted from the string named ALPHABET. If the list of letters in ALPHABET has been exhausted, control is transferred back to L1. At L1 the contents of ALPHABET are restored, SIZE is decremented again, and the sort continues on the next column.

```
LIST = LIST  $PIT
```

Add the words in the "pocket" onto LIST.

```
$PIT =  /(L6)
```

Set the "pocket" equal to the null string and transfer back to L6 to get the name of the next "pocket."

```
FIN SYSPOT= @ THE ALPHABETIZED LIST IS - @ LIST
```

After the sort is complete, that is, SIZE also reaches a minus one, this statement types the alphabetized list.

```
END  BEGIN
```

This statement performed the functions of both the END and STOP statement on FORTRAN. It indicates to physical end of the source deck for the interpreter. If control is transferred to it, either by a branch to END or, as in this case, by being executed as the next sequential instruction, execution of the program is ended. BEGIN is the statement label of the first instruction to be executed in the program. If this name is omitted, execution starts with the first statement, as in FORTRAN.

VIII. Other Features.

The language has several other features:

Recursive Functions
Balanced (with respect to parentheses) Fillers
Back Referencing (if the contents in a Filler match further down)
Tracing Capabilities (sense switch 1)
Memory Dump (on a control card).

NOTE: SNOBOL has been accepted by the COMMON library under the number 1.4.024.

```
##XEQ SNOBOL
BEGIN   SYSPIT          *SIZE*  @ @
START   SYSPIT          *WORDS* @ @                                 /F(L0)
        LIST            = LIST WORDS                                /(START)
L0      SYSPOT          = @THE LIST TO BE ALPHABETIZED IS -   @  LIST
        SYSPOT           =  VIOD
L1      ALPHABET        =  @ABCDEFGHIJKLMNOPQRSTUVWXYZ@
L2      SIZE             =  SIZE - @1@
        SIZE            @-@                                         /S(FIN)
L3      LIST            *WORD*  @,@  =                              /F(L5)
        WORD            *HEAD/SIZE*  *PIT/@1@*                      /F(L4)
        $PIT            =  $PIT     WORD   @,@                       /(L3)
L4       BIN             = BIN WORD  @,@                            /(L3)
L5      BIN             *LIST*  =
L6      ALPHABET        *PIT/@1@*        =      /F(L1)
        LIST            =  LIST  $PIT
        $PIT            =                                           /(L6)
FIN      SYSPOT       =   @THE  ALPHABETIZED LIST IS  -   @   LIST
END  BEGIN
9
ARMY,TEST,GLOBAL,ARMORY,GLOBE,ARM,TENSOR,ALIBI,ARE,GLOW,TENSE,TOTAL,CANCEL,
TONSIL,GLADIATOR,MOBILE,MOTILE,ANY,TORSION,PLATITUDE,FUMBLE,
CARD.WITH.NO.BLANKS.FOR.TRAILER.....................................
```

An Interpretive Input Routine for Linear Programming

D. J. Aigner
University of Illinois

1. Introduction

The LP system to be described below derived from a need to make available to business and economics students with essentially no programming background a set of easy-to-use codes for various tools in the fields of operations research and statistical analysis.

While as a general proposition it would seem that business schools are requiring increased mathematical and statistical sophistication on the part of students (as reflected in their curricula), a basic introduction to digital machines, required of all undergraduates, is not yet such a general reality. When such a course is a requirement in these curricula it will likely become of lesser importance - indeed, more difficult to justify economically the effort required in many cases to produce truly simple input codes for students' use, such as the one which is the subject of this note.

Of course the programming model, whether it be of linear or non-linear form, and as one of a number of such examples, finds application in fields other than business and economics. Thus viewed in a more global perspective the central University computing facilities, at least in the short run, should be concerned if not committed to providing a basic kit of popular tools which is easily accessible to the unsophisticated user. This last statement is not an apology for the existence of such people, but rather a judgment based on an observation of reality. The

educational process does not merely apply to the person who is listed as a student in the University records, but to his teachers as well. And while we usually speak of pedagogy as a teacher to student causal relation, it may as well be applied across disciplines on a teacher to teacher basis, and, as is becoming more apparent, the learning process, especially with regard to computer applications in "virgin" areas, also takes the form of a student to teacher feedback.

While this introduction may not seem appropos the more technical content to follow, it does establish, if its basic premise is agreed on, a justification for the cost of preparing rather complex (from a coding standpoint) I/O routines for many well-used tools of analysis.

At Illinois the premise in question is accepted, with the result that an integrated set of 37 programs for statistical techniques, matrix editing and manipulations, data transformation, and certain operations research tools has been produced; the programs are available individually or in combination through a single data input routine. This package, designated SSUPAC, is presently stored on the 7094 disk file and may be obtained by the user via the main operating system just as he would any compiler.

The linear programming code in SSUPAC, written originally by Glenn Rosbrook of the University's Statistical Service Unit, is unique not for reason that the code is superior in efficiency, generality, or flexibility to other codes, but because of its input routine. Indeed, any LP code could be used for calculation purposes, receiving its input from this routine. The input code has also been adapted as ALPS, A Linear Programming System on the College of Business Administration's 1620, and is used primarily as a pedagogical device for students in operations research courses.

2. Description of the Routine

The basic concept upon which ALPS is based is quite simple: to produce a code for linear programming which requires input of the simplest form, but is general enough to allow a variety of data modes (mixed perhaps), has very few restrictions for card punching, and will itself take care of adding slack or artificial variables where necessary. The result produced in attempting to meet these goals takes the form of an algebraic statement of the LP problem, e.g.

$$\begin{array}{ll} \text{Maximize} \\ \text{(or minimize)} \end{array} \qquad Z = x_1c_1 + x_2c_2 + \ldots + x_nc_n$$

Subject to

$$\begin{array}{l} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \left\{\begin{array}{l}\leq\\ =\\ \geq\end{array}\right. b_1 \\ \qquad\vdots \qquad\qquad\qquad\qquad\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \left\{\begin{array}{l}\leq\\ =\\ \geq\end{array}\right. b_m \end{array}$$

$$x_i \geq 0 \qquad i = 1, \ldots, n$$

translated ver batim onto punched cards, except for the non-negative restrictions. For example, the problem

$$\text{Maximize} \quad Z = .5x_1 + 6x_2 - 5x_3$$

Subject to

$$4x_1 + 6x_2 + 3x_3 = 24.5$$

$$x_1 + 1.5x_2 + 3x_3 \leq 12$$

$$3x_1 + x_2 \leq 12$$

could be inputed to ALPS as:

```
MAXIMIZE Z  =  .5X(1) + 6X(2) - 5X(3),
SUBJECT TO
CONSTRAINT (1) 4X(1) + 6X(2) + 3X(3)  =  24.5,
CONSTRAINT (2)  X(1) + 1.5E00X(2) + 3X(3) LE 12,
CONSTRAINT (3) 3X(1) +  X(2) LE 12,
END
```

Basically, the routine keys off the beginning letter of each statement, M, S, C, or E to determine the type of statement: "M" for the objective function (maximize or minimize), "S" for any editorial statement, "C" for a constraint statement, and "E" for the end of problem statement. In the objective and constraint statements an ending comma signals termination of the statement. Editorial statements can be used anywhere in the input, and statements may be punched anywhere on the card or cards necessary; continuation is automatic until the terminating comma is found and all blanks are ignored.

Coefficients may take on four "modes": integer as in 3X(1), Fortran "F" format as in .5X(1), floating as in 1.5E00X(2), and no coefficient as in X(1), which is interpreted as 1X(1). There are no restrictions on mixing modes in any statement. Forms of the constraint inequations are indicated by LE ($\leq$), GE ($\geq$), and E or = . Identical comments as above apply to the mode representation of the constraint constants.

As presently written, ALPS does error checking in the statements for invalid characters, integer subscripts, beginning letters and ending commas (where applicable) for all statements, etc. Also included is a capacity check for the maximum matrix size allowed, which of course varies according to the particular machine at hand.

ALPS prepares a data matrix from the above input by adding necessary slack and artificial variables with appropriate weights for the objective function, provides an initial basis, and, if required, transforms a minimization problem into a maximization problem as input to any LP solver. The basic process is outlined in Figure 1.

3. Coordination with a Solution Routine

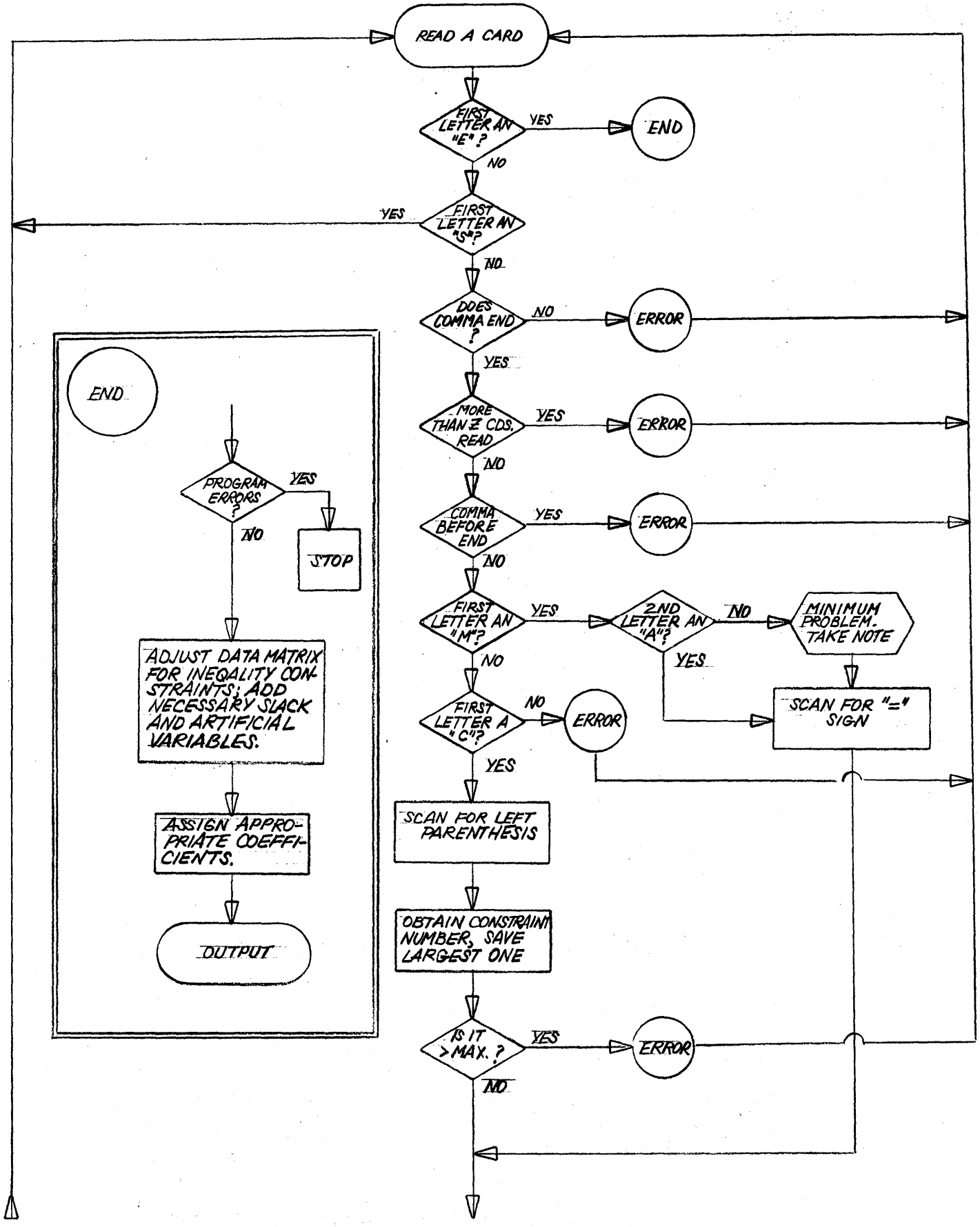
Bootstrapping the output of ALPS to a given calculation routine is obviously dependent in form on the hardware at hand. For small equipment without peripheral storage devices the most efficient method for coordination likely would be to prepare the data matrix (size determined by the solution routine and the hardware) in high-order memory and load the solving routine directly after ALPS puts the matrix into its proper addresses. With less imposing memory limitations, or possession of disk or tape storage, the coordination problem could be handled in a variety of ways, depending on the maximum problem size desired.

As presently written ALPS requires approximately 7,200 BCD positions plus the output data matrix* or about 1,000 words plus on a Boolean machine. Input statements are processed individually, then immediately translated into their appropriate locations in the data matrix.

Both operating versions at Illinois use a Fortran-coded revised simplex algorithm for computation.

---

* Or matrices, depending on the algorithm used.


READ A CARD
FIRST LETTER AN "E"?
YES
END
NO
FIRST LETTER AN "S"?
YES
NO
DOES COMMA END ?
NO
ERROR
YES
MORE THAN 2 CDS. READ
YES
ERROR
NO
COMMA BEFORE END
YES
ERROR
NO
FIRST LETTER AN "M"?
YES
2ND LETTER AN "A"?
NO
MINIMUM PROBLEM. TAKE NOTE
YES
SCAN FOR "=" SIGN
NO
FIRST LETTER A "C"?
NO
ERROR
YES
SCAN FOR LEFT PARENTHESIS
OBTAIN CONSTRAINT NUMBER, SAVE LARGEST ONE
IS IT > MAX. ?
YES
ERROR
NO
END
PROGRAM ERRORS ?
YES
STOP
NO
ADJUST DATA MATRIX FOR INEQALITY CONSTRAINTS; ADD NECESSARY SLACK AND ARTIFICIAL VARIABLES.
ASSIGN APPROPRIATE COEFFICIENTS.
OUTPUT

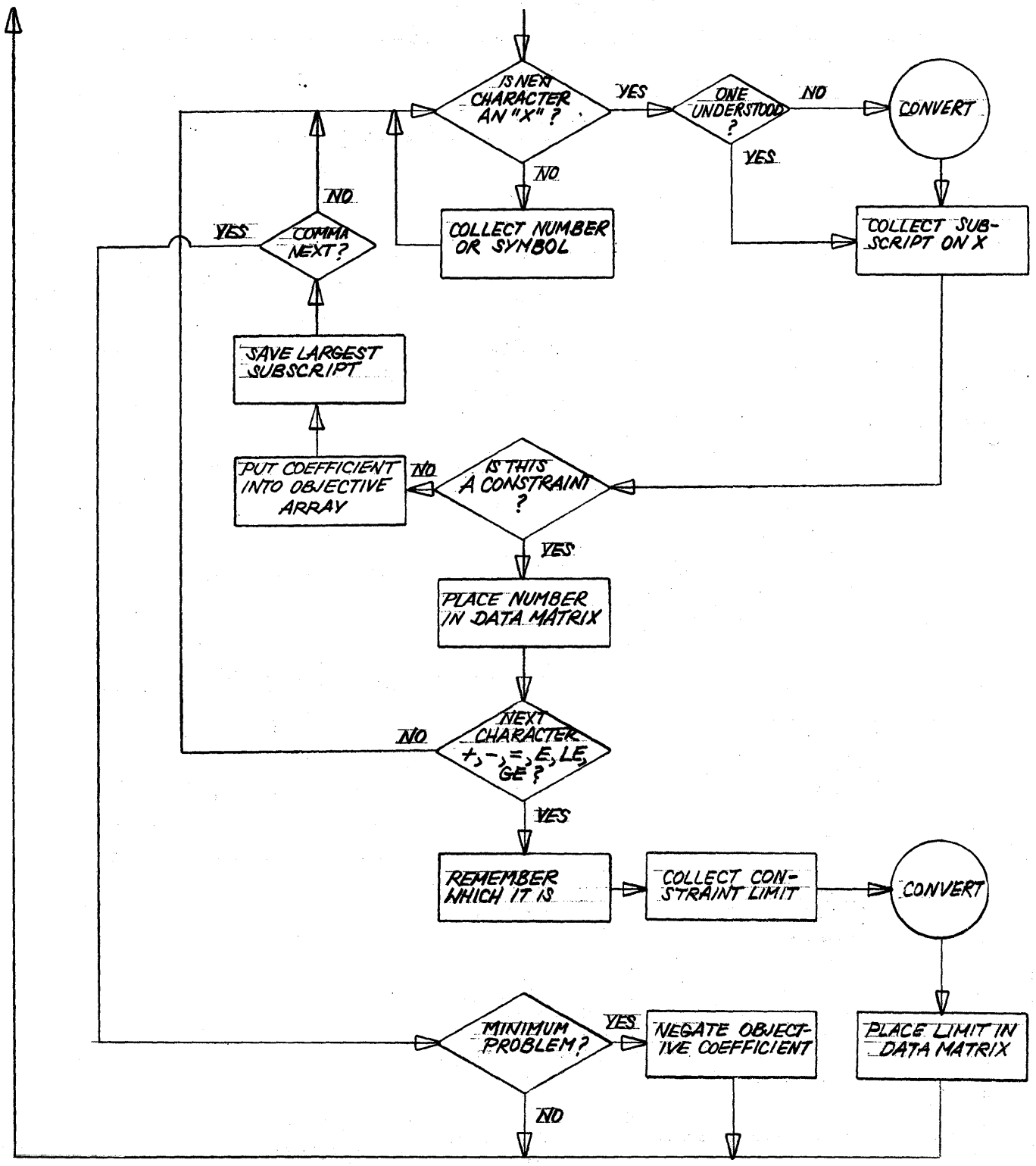


FIGURE 1. SKETCH FLOWCHART — ALPS

Appendix

Adaptations for the 1620

ALPS is now used in conjunction with a mag tape 1620, where the data matrix is outputed from ALPS to tape, the Fortran-coded LP solver is read into memory, and finally the data matrix is returned to memory from tape. A disk-oriented 1620 could be utilized in the same fashion, so that no intermediate output need be handled.

For a strictly card-oriented machine, it is possible to alleviate intermediate handling problems also, by preparing the data matrix in addresses appropriate to a given solution routine, and then loading the solution routine over ALPS. The basic modifications necessary to accomplish this consist of the following (references refer to SPS source listing):

Page	Change	To
17	EXIT	H B* - 12
18	RWT2	remove
18 - 19	TFM YECH+6, NUM through MES2 DAC 38	RNCD 0 B 0
21	BOX DSB 10, 21,, CONTRI DSB 10, 74,, CONST DSB 10, 1575,,	These locations must have their addresses set so as to place the arrays in their corresponding solution routine locations
21	BLAST DC 1'	Can be set to correspond to given memory capacity

The Fortran code used at present with ALPS is listed on pp. 22-24. On a 40K machine the maximum matrix size (coefficients plus constraint constants) is $n + m \leq 154$ with the normal significance levels $f = 8$, $k = 5$, maximum problem size being dependent upon the number of LE's and GE's included. The maximum problem size of 74 variables and 21 constraints could only occur if all constraints were equalities. Of course there are ways, given tapes or disks, to increase problem size by some "ping-pong" method, but our user's requirements hardly justify this. It is also difficult to justify using the 1620 for large LP problems given that one has two large-scale machines available for general use.

A copy of the user's information sheet for ALPS is also included to show examples of input and output for the LP package.

```
*  COMPACTION ROUTINE. AT STORE. ADDR GIVES RECORD MARK POSITION
       CF   ROUT
       RCTY
       RCTY
       TFM  CNUM.CONTRI
       TFL  -CNUM.ZERO
       AM   CNUM.10
       CM   CNUM.CONTRI+16490
       BN   *-36
       SF   START
       TFM  ADDR.CREC
       TDM  -ADDR.0
       AM   ADDR.1
       CM   ADDR.CREC+21
       BNE  *-36
       TFM  ADDR.CONST+15741
       TD   -ADDR.RMARK
       TFM  CDSAVE.0
       TFM  SAVE.0
       TFM  SAVE1.0
       TFM  NUM+3.0.7
AGAIN  TFM  COUNT.NUM
       TDM  EQSN+35.3.. TO LOOK FOR . SIGN
       TFM  ADDR.OVERLP
       TDM  -ADDR.0
       AM   ADDR.1
       CM   ADDR.OVERLP+80
       BNE  *-36
       TFM  ADDR.STCOL
       TDM  -ADDR.0
       AM   ADDR.1
       CM   ADDR.STCOL+159
       BNE  *-36
       TFM  CD.0
       SM   CD.1
       CF   C
       CF   ERR1...NEW SEQUENCE--COMMA CHECK
       SF   EQSN
       TFM  ADDR.NUM
START  TFM  COLM.STCOL
       RACDSTCOL
       SF   CHK...A CARD HAS JUST BEEM READ
       AM   CD.1
       AM   CDSAVE.1
       TD   WORK.STCOL+158
       TD   WORK-1.STCOL+157
       SF   WORK-1
       CM   WORK.0.10.SEE IF 80 IS BLANK
       BNE  ERR11
       TF   STCOL+158.RMARK
       BNF  *+24.START
       TFM  STCOL-2.71.10. SKIPS PAGE IN FORMAT CONTROL
       WACDSTCOL-2
       CF   START
       TFM  STCOL-2.00.10
* CHECK FOR END OF CARD
CHK    CM   COLM.STCOL+158
       BE   YES
       TD   WORK.-COLM
       SM   COLM.1
       TD   WORK-1.-COLM
```

```
        AM   COLM,1
        SF   WORK-1
        CM   ADDR,NUM+2
        BNE  *+72
        SF   NUM-1,
        CM   NUM,45,10, SEE IF END STATEMENT
        BE   END
        CM   NUM,62,10, SEE IF AN S
        BE   AGAIN
* CHECK FOR BLANK
        CM   WORK,00,10
        BNE  TSMITD
        AM   COLM,2
        B    CHK
TSMITDBNF  RND,CHK,,COMMA CHECK
        CF   CHK
        BNF  RND,ERR1
        TD   WORK,-COLM
        SM   COLM,1
        TD   WORK-1,-COLM
        AM   COLM,1
        SF   WORK-1
        CM   WORK,54,10,SEE IF AN M
        BE   ERR10
        CM   WORK,43,10,SEE IF A C
        BE   ERR10
        CM   WORK,62,10,SEE IF AN S
        BE   ERR10
        CM   WORK,45,10, SEE IF END STATEMENT
        BE   ERR10
RND     TD   -ADDR,-COLM
        SF   ERR1
        SM   ADDR,1
        SM   COLM,1
        TD   -ADDR,-COLM
        CF   -ADDR
        AM   ADDR,1
        AM   COLM,3
* CHECK FOR COMMA
        AM   ADDR,2
        CM   WORK,23,10
        BE   STIP
        B    CHK
* CHECK FOR FIFTH CARD
YES     CM   CD,5
        BE   ERR1
        B    START
ERR1    WATYNO1
        BTM  ROUT,0,10
        B    AGAIN
STIP    SM   ADDR,2
        TF   -ADDR,RMARK
*   CHECK FOR MATERIAL TO RIGHT OF A COMMA
        AM   COLM,2
COM     CM   COLM,STCOL+155
        BNN  RUNE
        TD   WORK-1,-COLM
        AM   COLM,1
        TD   WORK,-COLM
        AM   COLM,1
        SF   WORK-1
```

VON - SEE PAGE FOR ADDITIONAL SPS INSTRUCTIONS

```
        CM   WORK,00,10,TO SEE IF BLANK
        BE   COM
        WATY MES12
        BTM ROUT,0,10
        B    AGAIN
RUNE    NOP
*   INTERROGATION OF MAX, MIN, OR CONSTRAINT
        TFM CNUM,0
        CF   AMINI
        SF   EQSN
        TFM COUNT1,NUM
        TD   WORK-1,NUM-1
        TD   WORK,NUM
        SF   WORK-1
        CF   WORK
        CM   WORK,54,10,SEE IF AN M
        BE   M
        CM   WORK,43,10,SEE IF A C
        BE   C
        WATYERR4
        BTM ROUT,0,10
        B    AGAIN
M       TD   WORK,NUM+2
        TDM WORK-1,0,11
        CM   WORK,1,10, SEEIF AN A
        BE   EQSN-12
AMINI   SF   AMINI
        B    EQSN-12
C       SF   C
        TDM WORK-1,0,11
        TDM EQSN+35,2,,TO LOOK FOR A LEFT PAREN RATHER THAN = SIGN
        AM   COUNT1,1
EQSN    AM   COUNT1,2
        TD   WORK,-COUNT1
        CM   WORK,3,10, SEE IF = SIGN
        BE   *+24
        B    EQSN
*   THE ABOVE LEAVES COUNT1 AT = SIGN, LEFT SIDE
        AM   COUNT1,2
        BNF MAX,C
*   FLAG AT C IF A CONSTRAINT CARD
*   FILL IN CONSTRAINT ROUTINE, COUNT1 IS ON LEFT SIDE OF FIRST NO, IN PAREN
        CF   FIX
        AM   COUNT1,1
        TD   CNUM,-COUNT1
        AM   COUNT1,1
        TD   WORK,-COUNT1
        CM   WORK,7,10, TO SEE IF IT IS A 2-PLACE SUBCRIPT
        BNE *+60
        AM   COUNT1,1
        TD   CNUM-1,CNUM
        TD   CNUM,-COUNT1
        AM   COUNT1,1
*   COUNLD PUT A CHECK IN HERE FOR RIGHT SUBSCRIPT
        CM   CNUM,21
        BNP *+36
        WATYERR8
        BTM ROUT,0,10
        AM   COUNT1,2
*    SAVE1 WILL CONTAIN THE LARGEST SUBSCRIPT
        C    SAVE1,CNUM
```

```
        BP   *+24
        TF   SAVE1,CNUM
MAX     TF   COUNT2,COUNT1
        TFM  COUNT,FORM-31
        TD   WORK,-COUNT1
        AM   COUNT1,1
        TD   WORK-1,-COUNT1
        SF   WORK-1
        SM   COUNT1,1
        CM   WORK,76,10, SEE IF AN X
        TDM  WORK-1,0,11
        BNE  RL
        BNF  PAREN-12,EQSN
        TFL  EX,ONE
        SM   COUNT1,2
        TD   WORK,-COUNT1
        CM   WORK,2,10, TO SEE IF NEGATIVE ONE
        BNE  *+24
        SF   EX-2
        AM   COUNT1,2
        B    PAREN
*  CONDITION TO NOT SET A NUMBER TO ONE
RL      TD   WORK,-COUNT1
        CM   WORK,7,10, CLEAR FLAG IF A NUMBER THE FIRST TIME
        BNE  *+24
        CF   EQSN
        TD   -COUNT,-COUNT1
        AM   COUNT,1
        AM   COUNT1,1
        TD   -COUNT,-COUNT1
        AM   COUNT,1
        AM   COUNT1,1
        B    MAX+24
        BTM  FORM,0,10
PAREN   SF   EQSN
        TDM  WORK-1,0,11
        TFM  PLACE,0
        AM   COUNT1,4
        TD   WORK,-COUNT1
        CM   WORK,7,10, SEE IF IT IS A NUMBER
        BE   *+48
        WATYERR5
        BTM  ROUT,0,10
        B    AGAIN
        AM   COUNT1,1
        TD   PLACE,-COUNT1
        AM   COUNT1,1
        TD   WORK,-COUNT1
        CM   WORK,0,10, SEE IF RIGHT PAREN
        BE   DESERT
        CM   WORK,7,10,DEE IF A NUMBER
        BE   *+48
        WATYERR5
        BTM  ROUT,0,10
        B    AGAIN
        TD   PLACE-1,PLACE
        AM   COUNT1,1
        TD   PLACE,-COUNT1
        AM   COUNT1,1
        TD   WORK,-COUNT1
        CM   WORK,0,10,  SEE IF RIGHT PAREN
```

```
       BE   *+48
       WATYERR5
       BTM ROUT,0,10
       B    AGAIN
DESERTBNF MPLACE,C
*   CNUM IS PRESENT CONSTRAINT NUMBER, PLACE IS  X(PLACE),
       TF   ADDR,CNUM
       SM   ADDR,1
       MM   ADDR,750,9
       SF   95
       TF   ADDR,99
       MM   PLACE,10
       SF   95
       SM   99,10
       A    ADDR,99
       AM   ADDR,CONST
       TFL  -ADDR,EX
       B    EST
MPLACESM   PLACE,1
       TF   ADDR,PLACE
       MM   ADDR,10
       SF   95
       TF   ADDR,99
       AM   ADDR,CONTRI
       TFL  -ADDR,EX
       AM   PLACE,1
       C    SAVE,PLACE
       BNN  *+24
       TF   SAVE,PLACE,,SAVE WILL CONTAIN THE LARGEST SUBSCRIPT
*   CHECK FOR TWO EQUAL SUBSCRIPTS
       AM   PLACE,OVERLP
       BNR  *+36,-PLACE
       WATYERR6
       BTM ROUT,0,10
       TD   -PLACE,RMARK
EST    SF   EQSN
       AM   COUNT1,2,,LEAVES IT ON LHS OF FIRST DIGIT OF NEXT NO,
       TF   COUNT2,COUNT1
       AM   COUNT2,1
       BNF  DPCONT-12,C
       TD   WORK,-COUNT1
       TF   ADDR,CNUM
       SM   ADDR,1
       AM   ADDR,CREC
       CM   WORK,01,10, SEE IF A PLUS SIGN
       BE   MAX
       CM   WORK,02,10, SEE IF A MINUS SIGN
       BE   MAX
       CM   WORK,03,10, SEE IF = SIGN
       BE   FIX
       CM   WORK,05,10, SEE IF  LE
       BNE  *+48
       AM   COUNT1,2
       TD   -ADDR,RMARK
       B    FIX
       CM   WORK,04,10, SEE IF  E  OR  GE
       BNE  NO7
       TD   WORK,-COUNT2
       CM   WORK,5,10, SEE IF E
       BE   FIX
       TDM -ADDR,0,11, SET A FLAG FC
```

```
        AM   COUNT1,2
FIX     TFM  COUNT,FORM-31
        AM   COUNT1,2,,AT LHS OF FIRST NO. AFTER C-CONDITION
        TD   WORK,-COUNT1
        CM   WORK,7,10, SEE IF A NO.
        BE   LLL-12
        CM   WORK,0,10, SEE IF A PERIOD
        BE   LLL-12
        CM   WORK,2,10, SEE IF A NEG. NO.
        BNE  *+24
        SF   FIX,,, CHANGE SIGNS ALL NOS. JUST ENTERED IF A FLAF AT FIX
        AM   COUNT1,2
        TF   COUNT2,COUNT1
LLL     TF   COUNT3,COUNT1
        AM   COUNT3,1
        BNR  *+24,-COUNT3
        B    FIX2
        TD   -COUNT,-COUNT1
        AM   COUNT,1
        TD   -COUNT,-COUNT3
        AM   COUNT1,2
        AM   COUNT,1
        B    LLL
FIX2    BTM  FORM,0,10
        TF   ADDR,CNUM
        SM   ADDR,1
        MM   ADDR,10,10
        SF   95
        TF   ADDR,99
        AM   ADDR,BOX
        TFL  -ADDR,EX
* CHECK TO SEE IF NEG. CONSTRAINT
        BNF  AGAIN,FIX
        TF   ADDR,CNUM
        SM   ADDR,1
        MM   ADDR,750,9
        SF   95
        TF   ADDR,99
        AM   ADDR,CONST
        SM   ADDR,2
        TF   PLACE3,ADDR
        AM   PLACE3,740
ZAP     BNF  *+36,-ADDR
        CF   -ADDR
        B    *+24
        SF   -ADDR
        C    ADDR,PLACE3
        BE   ZIP
        AM   ADDR,10
        B    ZAP
ZIP     TF   ADDR,CNUM
        SM   ADDR,1
        AM   ADDR,CREC
        BNR  *+48,-ADDR
        BNF  AGAIN,-ADDR
        TDM  -ADDR,0,11
        BNF  AGAIN,-ADDR
        TD   -ADDR,RMARK
        B    AGAIN
        BNR  MAX,-COUNT2
DPCONTBNF  AGAIN,AMINI
```

```
*  MINIMIZER
       TFM  COUNT2,0
       TFM  ADDR,CONTRI-2
       BNF  *+36,-ADDR
       CF   -ADDR
       B    *+24
       SF   -ADDR
       AM   COUNT2,1
       AM   ADDR,10
       C    SAVE,COUNT2
       BNE  *-84
       B    AGAIN
*  CONVERSION ROUTINE
       DC   31,0
FORM   SF   FLAG
       CF   E
       CF   FLAG2
       CF   MINUS
       CF   MINUSE
       TFM  EX,0,10
       TFM  EX-2,0
       TFM  EX-5,0
       TFM  AT,EX-9
       TFM  COUNT,FORM-31
       CF   FORM-31
TRY    TD   WORK,-COUNT
       TDM  WORK-1,0,11
       CM   COUNT,FORM-1
       BE   SERVE
       CM   WORK,7,10,SEE IF A DIGIT
       BE   DIGIT
       TD   WORK-1,-COUNT
       AM   COUNT,1
       TD   WORK,-COUNT
       SM   COUNT,1
       SF   WORK-1
       CM   WORK,45,10,SEE IF AN E
       BE   E
       CM   WORK,03,10,SEE IF A PERIOD
       BE   PERIOD
       CM   WORK,10,10,SEE IF A PLUS SIN
       BNE  *+36
       AM   COUNT2,2
       B    PERIOD-24
       CM   WORK,20,10,SEE IF A MINUS SIGN
       BNE  *+36
       AM   COUNT2,2
       B    MINUS
       CM   WORK,0,10,SEE IF A BLANK
       BE   SERVE
       WATYERR2
       RCTY
       WATY MES6
       WNTYCDSAVE-3
       RCTY
       RCTY
       TFM  *+18,FORM-31
       TDM  FORM-31,0,,TO SET TO ZER/
       AM   *-6,1
       CM   *-18,FORM-2
       BNE  *-36
```

```
        SF   ROUT
        B    AGAIN
DIGIT   AM   COUNT,1
        TD   WORK,-COUNT
        TDM  WORK-1,0,11
        SM   COUNT,1
        CM   WORK,0,10,SEE IF A ZERO
        BNE  ANUM
FLAG    BNF  ANUM,FLAG
FLAG2   BNF  *+24,FLAG2
        SM   EX,1,10,HAVE GONE BY . W/O NON-ZERO DIGIT
        B    PERIOD-24
ANUM    AM   COUNT,1
        TD   -AT,-COUNT
        SM   COUNT,1
        BNF  *+24,FLAG2,,, . HAS NOT BEEN FOUND,EXP IS NOT SET
        B    *+24
        AM   EX,1,10
        CF   FLAG
        AM   AT,1
        AM   COUNT,2
        B    TRY
PERIODAM   COUNT,1
        TD   WORK,-COUNT
        TDM  WORK-1,0,11
        CM   WORK,0,10, SEE IF NUMBER IS THRU
        BE   SERVE
*   COUND PUT IN CHECK FOR PERIOD, AM AT RHS OF THING STARTING W/ BLANK
        SM   COUNT,1
        SF   FLAG2
        B    PERIOD-24
MINUS   SF   MINUS
        B    PERIOD-24
MINUSESF   MINUSE
E       AM   COUNT,2
        SF   E
        TDM  WORK-1,0,11
        TD   WORK,-COUNT
        CM   WORK,1,10,SEE IF A PLUS SIGN
        BE   E
        CM   WORK,2,10,SEE IF A MINUS SIGN
        BE   MINUSE
        AM   COUNT,1
        TD   WORK-1,-COUNT
        AM   COUNT,2
        TD   WORK,-COUNT
        SF   WORK-1
        BNF  *+24,MINUSE
        SF   WORK
        A    EX,WORK
SERVE   SF   EX-1
        SF   EX-9
        CF   EX-2
        BNF  *+24,MINUS
        SF   EX-2
        TFM  *+18,FORM-31
        TDM  FORM-31,0,,TO SET TO ZERO
        AM   *-6,1
        CM   *-18,FORM-2
        BNE  *-36
*   FOR I-CONVERSION
```

```
        BNF *+24,E
        BB
        BNF *+24,FLAG2
        BB
        AM  COUNT2,1
*   COUNT2 IS ON RHS OF FIRST NO.
        TF  PLACE,COUNT1
        S   PLACE,COUNT2
        TFM ADDR,0
        SM  PLACE,2
        AM  ADDR,1
        CM  PLACE,0
        BH  *-36
        SF  ADDR-1
        TF  EX,ADDR
        BB
*    END ROUTINE
END     NOP
        BNF BRND,ROUT,,IF ERRORS EXIST, EXIT
        EXIT
BRND    TFM PLACE1,CREC,,,PLACE1 CHECKS FOR C-CONDITION
        TFM J,0,,, INDICATES USE OF EXTRA COLUMNS
        TFM SLIP,CONTRI
        MM  SAVE,10,10
        SF  95
        A   SLIP,99
        MM  SAVE1,10,10
        SF  95
        A   SLIP,99,,SLIP IS BELOW ADDITIONAL COLUMN
        TFM JJ,0
        TFM I,0
*        FOR I MATRIX, PLACE3
        TF  PLACE3,SAVE
        A   PLACE3,I
        MM  PLACE3,10,10
        SF  95
        TF  PLACE3,99
        MM  I,750,9
        SF  95
        A   PLACE3,99
        AM  PLACE3,CONST
LOOP    BNF RECK,-PLACE1
*   GE ROUTINE
        TFL -PLACE3,NEGONE
        TFL -SLIP,LGNEG
        AM  SLIP,10
*      FOR EXTRA COLUMNS, PLACE2
        TF  PLACE2,SAVE
        A   PLACE2,SAVE1
        A   PLACE2,J
        MM  PLACE2,10,10
        SF  95
        TF  PLACE2,99
        MM  I,750,9
        SF  95
        A   PLACE2,99
        AM  PLACE2,CONST
        TFL -PLACE2,ONE
        AM  J,1
        B   INCRE,,,,TO INCREMENT PLACE 1 AND 3 AND I
RECK    BNR NONE,-PLACE1
```

```
* LE ROUTINE
        TFL  -PLACE3,ONE
        B    INCRE
NONE    TFL  -PLACE3,ONE
        TF   EQDWN,I
        A    EQDWN,SAVE
        MM   EQDWN,10,10
        SF   95
        TF   EQDWN,99
        AM   EQDWN,CONTRI
        TFL  -EQDWN,LGNEG
INCRE   AM   I,1
        AM   PLACE1,1
        AM   PLACE3,760
        C    SAVE1,I
        BNE  LOOP
*     PUT BOX AWAY AT END OF CONSTRAINTS
        TFM  PLACE1,BOX
        TF   PLACE2,SAVE
        A    PLACE2,SAVE1
        A    PLACE2,J
        MM   PLACE2,10,10
        SF   95
        TF   PLACE2,99
        AM   PLACE2,CONST
        TFM  I,0
NIFTY   TFL  -PLACE2,-PLACE1
        AM   PLACE1,10
        AM   PLACE2,750
        AM   I,1
        C    SAVE1,I
        BNE  NIFTY
        TF   SUB,SAVE
        A    SUB,SAVE1
        A    SUB,J
        RWT2
        TFM  TRT+6,NUM+30
TRT     TFM  NUM+30,0
        CM   TRT+6,NUM+160
        BE   *+36
        AM   TRT+6,5
        B    TRT
        TF   NUM+4,SUB
        TF   NUM+9,SAVE1
        TF   NUM+14,SAVE
        CF   NUM
        CF   NUM+5
        CF   NUM+10
        TF   NUM+80,RMARK
        TFM  YECH+6,NUM
        BTM  WRITE,0
        TD   SLIP,CONST-9
        TD   CONST-9,RMARK
        TFM  YECH+6,CONTRI-9
        BTM  WRITE,0
        TD   CONST-9,SLIP
        SF   CONST-9
        TFM  YECH+6,CONST-9
        BTM  WRITE,0
        RWT2
        EXIT
```

```
       H
       DC  5,0
WRITE  TFM CD,5,10
       TFM COUNT,5,10
       BV  *+12
       CTI2
YECH   WNT20
       BNWC*+14
       BKT242000
       SM  COUNT,1,10
       BNZ WRITE+24
       RCTY
       WATYMES1
       EFT2
       SM  CD,1,10
       BNZ WRITE+12
       RCTY
       WATYMES2
       H
       B   WRITE
MES1   DAC 17,TAPE WRITE CHECK'
MES2   DAC 38,PUT NEW TAPE OF TWO, THEN PRESS START'
NO1    DAC 48,USED MORE THAN FIVE CARDS FOR ONE SPECIFICATION'
ERR2   DAC 29,INVALID CHARACTER IN A NUMBER
       DAC 1,'
ERR4   DAC 33,FIRST LETTER NOT A C, M, E, OR S'
ERR5   DAC 18,INVALID SUBSCRIPT'
ERR6   DAC 47,TWO SUBSCRIPTS ARE EQUAL IN OBJECTIVE FUNCTION'
NO7    RCTY
       WATYERR7
       BTM ROUT,0,10
       B   AGAIN
ERR7   DAC 38,NO +, -, GE, LE, =, OR E AFTER AN X(I)
       DAC 1,'
ERR8   DAC 37,CONSTRAINT NUMBER IS GREATER THAN 21'
MES4   DAC 41,A COMMA DOES NOT TERMINATE THE END OF A C
       DAC 36,ONSTRAINT OR THE OBJECTIVE FUNCTION'
MES5   DAC 25,COLUMN 80 IS NOT A BLANK'
MES6   DAC 25,ERROR IS IN CARD NUMBER '
MES10  DAC 35,ERRORS NOT CHECKED IN CARD NUMBER '
MES12  DAC 45,A COMMA IS BEFORE THE END OF A SPECIFICATION'
       DC  5,0
ROUT   RCTY
       WATYMES6
       WNTYCDSAVE-3
       SF  ROUT
       RCTY
       RCTY
       BB
ERR11  WATYMES5
       BTM ROUT,0,10
       TF  STCOL+158,RMARK
       TFM STCOL-2,0,10
       WACDSTCOL-2
       B   AGAIN
ERR10  WATY MES4
       SM  CDSAVE,1
       BTM ROUT,0,10
       AM  CDSAVE,1
       CM  WORK,45,10,SEE IF AN END CARD
       BE  END
```

```
        WATYMES10
        WNTYCDSAVE-3
        RCTY
        B    AGAIN
        DC   2,0
STCOL   DAC  5,00000
        DC   50,0
        DC   50,0
        DC   50,0
RMARK   DC   2,'
        DC   8,10000000
ONE     DC   2,01
        DC   8,-50000000
LGNEG   DC   2,06
CNUM    DC   5,0
CD      DC   5,0
I       DC   5,0
J       DC   5,0
FINAL   DC   5,0
PLACE1  DC   5,0
PLACE2  DC   5,0
PLACE3  DC   5,0
SLIP    DC   5,0
EQDWN   DC   5,0
COUNT1  DC   5,0,,MOVING ADDRESS OF NUMBER
COUNT2  DC   5,0,,ADDRESS OF BEGINNING OF NUMBER
COUNT3  DC   5,0
CDSAVE  DC   5,0
        DC   1,'
AT      DC   5,0
JJ      DC   5,0
PLACE   DC   5,0
ADDR    DC   5,0
COLM    DC   5,0
COUNT   DC   5,0
        DC   8,0
EX      DC   2,0
WORK    DC   2,K0
NUM     DAC  3,000
        DC   50,0
        DC   50,0
        DC   50,0
        DC   50,0
        DC   50,0
        DC   50,0
        DC   50,0
        DC   50,0
        DC   50,0
        DC   50,0
        DC   50,0
        DC   50,0
        DC   50,0
        DC   50,0
        DC   50,0
        DC   50,0
        DC   50,0
        DC   50,0
        DC   8,0
ZERO    DC   2,-99
        DC   8,-10000000
NEGONE  DC   2,01
OVERLPDSC    50,0
        DC   32,0
```

```
BOX    DSB 10,21,,FOR PLACEMENT OF NUMBER FOLLOWING C-CONDITIONS
CREC   DSC 21,0,,FOR PLACEMENT OF CONSTRAINT CONDITIONS
       DC  5,0
SAVE   DC  5,0
SUB    DC  5,0
SAVE1  DC  5,0,, FOR THE NUMBER OF CONSTRAINTS
CONTRIDSB 10,74,,FOR STORAGE OF MAX-MIN VECTOR
CONST  DSB 10,1575,,FOR STORAGE OF CONSTRAINT MATRIX
BLAST  DC  1,'
       DEND402
```

```
      DIMENSION B(75,21),C(21),N(21),ZSTAR(75),P(74)
      COMMON N1
      COMMON B
C     MAXIMUM NUMBER OF CONSTRAINTS IS 21
C     NUMBER OF VARIABLES ALLOWED = 74-NUMBER OF CONSTRAINTS USED-NO. OF  GE'S
      READ INPUT TAPE 2,51,NCOL,NROW,NVAR
      READ TAPE 2,P
      PUNCH 101
      READ TAPE 2,B
      NT=0
 1111 IF(NT) 1188,1188,1199
 1199 PUNCH 110
 1188 LL=0
      NN=NCOL + 1
      Z = 0.0
      L2 = 1
   61 IF(NN-8)62,62,65
   65 LL = LL+8
      GO TO 68
   62 LL=LL+NN
      Z = 1.0
   68 PUNCH 20,(J,J=L2,LL)
      DO 40 I=1,NROW
   40 PUNCH 25,I,(B(J,I),J=L2,LL)
      IF (Z-1.0) 75,88,88
   75 NN=NN-8
      L2 = L2+8
      GO TO 61
   88 IF(NT) 1169,1169,1160
 1160 PUNCH 111
      GO TO 1165
 1169 PUNCH 103
 1165 LL=0
      NN=NCOL
      Z=0
      L2=1
  161 IF (NN-8)162,162,165
  165 LL=LL+8
      GO TO 168
  162 LL=LL+NN
      Z=1.
  168 PUNCH 20,(J,J=L2,LL)
      I=1
      IF(NT) 1220,1220,1222
 1222 PUNCH 25,I,(ZSTAR(J),J=L2,LL)
      GO TO 1221
 1220 PUNCH 25,I,(P(J),J=L2,LL)
 1221 IF(Z-1.)175,188,188
  175 NN=NN-8
      L2=L2+8
      GO TO 161
  188 IF(NT) 1270,1270,399
 1270 NN=NVAR+1
      II=0
      DO 1 J=NN,NCOL
      DO 1 I=1,NROW
      IF(B(J,I)-1.)1,9,1
    9 II=II+1
      N(II)=J
      C(II)=P(J)
    1 CONTINUE
```

```
      N1=0
 1501 LL=0
      I=1
      Z=0.
      L2=1
      NN=II
      IF(N1)1502,1502,1503
 1502 PUNCH 104
      GO TO  961
 1503 PUNCH 105
  961 IF(NN-8)962,962,965
  965 LL=LL+8
      GO TO 968
  962 LL=LL+NN
      Z=1.
  968 IF(N1)1504,1504,1505
 1504 PUNCH 20,(J,J=L2,LL)
      PUNCH 29,(N(J),J=L2,LL)
      GO TO 1506
 1505 PUNCH 20,(J,J=L2,LL)
      PUNCH 25,I,(C(J),J=L2,LL)
 1506 NN=NN-8
      L2=L2+8
      IF(Z-1.)961,988,988
  988 IF(N1)1507,1507,1508
 1507 N1=1
      GO TO 1501
 1508 L1=NCOL+1
  200 DO 203 I=1,NCOL
      ZSTAR(I)=-P(I)
      DO 203 J=1,NROW
  203 ZSTAR(I)=ZSTAR(I)+C(J)*B(I,J)
C     OPTIMALITY CHECK AND SIMPLEX CRITERION I
  400 FLAG=0
      SMALL=0
      DO 404 I=1,NCOL
      IF(ZSTAR(I))402,404,404
  402 FLAG=1.
      IF(ZSTAR(I)-SMALL)403,404,404
  403 SMALL=ZSTAR(I)
      IN=I
  404 CONTINUE
      IF(FLAG)900,900,405
C     APPLY SIMPLEX CRITERION II
  405 SMALL=9.E35
      DO 409 I=1,NROW
      IF(B(IN,I))409,409,408
  408 QUOT=B(L1,I)/B(IN,I)
      IF(QUOT -SMALL)411,410,409
  411 JN=I
      SMALL=QUOT
      GO TO 409
  410 IF (B(IN,I)-B(IN,JN))409,409,411
  409 CONTINUE
      IF(SMALL-9.E35)500,407,407
  407 PUNCH 1001
      GO TO  9089
C  NORMALIZE EQUATIONS WRT ENTERING VARIABLE
  500 DO 501 J=1,L1
  501 ZSTAR(J)=B(J,JN)/B(IN,JN)
      DO 502 I=1,NROW
```

```
      PROD=B(IN,I)
      DO 502 J=1,L1
  502 B(J,I)=B(J,I)-ZSTAR(J)*PROD
      DO 503 J=1,L1
  503 B(J,JN)=ZSTAR(J)
      C(JN)=P(IN)
      N(JN)=IN
      GO TO 200
C  OUTPUT
  900 NT=1
      GO TO 1111
  399 PUNCH 112
      Z=0.
      DO 909 I=1,NROW
      P(21)=B(L1,I)*C(I)
      Z=Z+P(21)
  909 PUNCH 120,     N(I),B(L1,I),C(I),P(21)
      PUNCH 113,Z
  113 FORMAT (27H0OPTIMAL FUNCTIONAL VALUE =   , F14.4 )
  120 FORMAT(5X,I5,6X,F14.4,9X,F14.4,2X,F14.4)
  110 FORMAT (17H1SOLUTION MATRIX   )
  111 FORMAT(35H2OPPORTUNTIY COSTS OR SHADOW PRICES )
  112 FORMAT(14H5VARIABLE USED,8X,8HQUANTITY,10X,13HCONTRIB./UNIT,
     111X,5HVALUE)
 1001 FORMAT(19H5SOLUTION UNBOUNDED )
   51 FORMAT(3I5)
  105 FORMAT(6H5COSTS )
  104 FORMAT(16H5BASIS VARIABLES )
  103 FORMAT(30H2CONTRIBUTION (PROFIT OR COST) )
   29 FORMAT(5X,5I14/1H9,26X,3I14)
   25 FORMAT (1X,I3,2X,5F14.4/1H9,27X,3F14.4)
   20 FORMAT (1H0,3X,5I14/1H9,25X,3I14)
  101 FORMAT(13H0MATRIX CHECK)
 9089 END
```

********** A LINEAR PROGRAMING SYSTEM (ALPS) **********

I. GENERAL INFORMATION

THE FUNCTION OF LINEAR PROGRAMING IS TO OPTIMIZE A LINEAR OBJECTIVE FUNCTION SUBJECT TO LINEAR CONSTRAINTS. FOR A DISCUSSION OF PROCEDURES AND INTERPRETATIONS SEE CHURCHMAN, ACKOFF, ARNOFF, INTRODUCTION TO OPERATIONS RESEARCH, NEW YORK, WILEY, 1961, OR ANY GENREALLY ACCEPTED TEXT ON THE SUBJECT. THIS PROGRAM USES THE GENERAL SIMPLEX METHOD. TO PRINT THE PUNCHED OUTPUT ON THE IBM 407, USE THE OUTPUT BOARD WITH SWITCHES ONE AND TWO DOWN (FOR FORMAT CONTROL).

II. STATEMENTS (IN ORDER)

A. CALL CARDS

1. $$JOB

COLUMNS 1-5 $$JOB
6 B (BLANK)
7-12 ID NO., STAFF, FAC, OR JOB NUMBER
13 B
14-33 NAME, LAST NAME FIRST (USE A COMMA)
34 B
35-38 COURSE NAME ABBREVIATION OR GENL IF NON-CLASS WORK
39 B
40-42 COURSE NUMBER, IF ANY
43 B
44-45 SECTION, IF ANY, RIGHT JUSTIFIED
46 B
47-48 PROBLEM NUMBER, IF ANY, RIGHT JUSTIFIED
49-80 B

2. $$ALPS

(IN COLUMNS 1-6)

B. OBJECTIVE FUNCTION

1. PUNCH MAXIMIZE OR MINIMIZE, FOLLOWED BY YOUR FAVORITE FUNCTIONAL NAME (OPTIONAL), AND THEN PUNCH AN EQUAL SIGN (=), FOLLOWED BY THE FUNCTION, AS SHOWN IN THE EXAMPLES. IT IS NEAT BUT NOT NECESSARY TO BEGIN PUNCHING IN COLUMN 1. ALL BLANKS ARE IGNORED.
2. FOLLOW IT WITH A COMMA. USE NO MORE THAN FIVE CARDS.

C. 'SUBJECT TO THE FOLLOWING' IS OPTIONAL. ACTUALLY, EDITORIAL COMMENTS MAY BE PLACED ANYWHERE BETWEEN SPECIFICATIONS (AFTER $$ALPS) BY SIMPLY HAVING AN S AS THE FIRST CHARACTER.

D. CONSTRAINTS

1. IT IS NEAT BUT NOT NECESSARY TO START IN COLUMN 1.
2. PUT THE CONSTRAINT NUMBER IN PARENTHESIS FOLLOWING THE WORD 'CONSTRAINT', THEN WRITE THE CONSTRAINT FUNCTION, FOLLOWED BY ONE OF THE SYMBOLS OF EQUALITY OR INEQUALITY.
   (A) LE MEANS LESS THAN OR EQUAL TO
   (B) GE MEANS GREATER THAN OR EQUAL TO
   (C) E OR = MEANS EQUAL TO
3. THEN WRITE THE CONSTRAINING FUNCTIONAL VALUE, IN ANY FORM ACCEPTABLE TO A COEFFICIENT (SEE BELOW), FOLLOWED BY A COMMA.

E. 'END' STATEMENT, PUNCHED IN ANY COLUMN, SIGNALS THE END OF DATA COMPILATION AND INITIATES EXECUTION.

III. RESTRICTIONS

A. THE MAXIMUM NUMBER OF CONSTRAINTS IS 21.

B. THE NUMBER OF VARIABLES ALLOWED (NOT INCLUDING SLACK OR ARTIFICIAL VARIABLES) IS BETWEEN 32 AND 74. TO BE MORE SPECIFIC, IT IS 74 MINUS THE NUMBER OF CONSTRAINTS USED MINUS THE NUMBER OF CONSTRAINTS USING THE NOTATION 'GE' (MEANING GREATER THAN OR EQUAL TO). THE FORMULA IS, USING NOTATION VERBALIZED IN THE LAST SENTENCE.

V = 74 - C'S - GE'S.

C. ONE SPECIFICATION (A CONSTRAINT OR THE OBJECTIVE FUNCTION) MUST NOT BE WRITTEN ON MORE THAN FIVE (5) CARDS.

D. A COMMA MUST FOLLOW EACH SPECIFICATION. THE END STATEMENT AND COMMENT STATEMENTS (BEGINNING WITH AN S) DO NOT REQUIRE THE TERMINATING COMMA.

E. WRITE NOTHING IN COLUMN 80.

F. USE OF NUMBERS FOR COEFFICIENTS--FOUR MODES POSSIBLE, AND THEY MAY ALL BE USED IN ANY ONE SPECIFICATION.

1. FIXED POINT MODE (AN INTEGER)
2. FLOATING POINT MODE (YOU SUPPLY THE DECIMAL POINT)
3. NO NUMBER IN FRONT OF AN X SIGNIFIES A 1, E.G., X(25) IS INTERPRETED AS 1X(25)
4. EXPONENTIAL MODE
   - A. NOT RECOMMENDED BECAUSE THE COEFFICIENTS SHOULD BE BETWEEN 1 AND 10,000, SAY, FOR ACCURATE RESULTS. IF THE COEFFICIENTS ARE NOT IN THIS RANGE, A SCALE FACTOR SHOULD BE USED BEFORE PUNCHING.
   - B. MAY USE FIXED OR FLOATING MANTISSA.
   - C. FOLLOW IT WITH AN E, THEN A MINUS OR PLUS SIGN (OPTIONAL), AND THEN A TWO DIGIT EXPONENT.
   - D. EXAMPLE-- 22E03 IS INTERPRETED AS 22,000.00

G. SUBSCRIPTS ARE PUT IN PARENTHESES FOLLOWING THE 'X'. THEY MUST NOT CONTAIN A DECIMAL POINT, AND ARE, OF COURSE, ONE OR TWO DIGITS IN LENGTH.

IV. ERROR STATEMENTS

A. ALPS HAS A LIMITED CAPACITY TO CHECK FOR ERRORS. THE MATRIX CHECK SHOULD ALWAYS BE VISUALLY COMPARED WITH YOUR INPUT DATA TO INSURE ACCURACY OF THE RESULTS. THE ERROR STATEMENTS ARE SELF-EXPLANITORY. ALL ERRORS EXCEPT THE FIRST TWO INHIBIT EXECUTION. THE MONITOR IS CALLED AFTER ALL CARDS ARE CHECKED FOR ERRORS. ONLY THE FIRST ERROR IN EACH CARD IS USUALLY GIVEN.

B. THEY ARE--

TAPE WRITE CHECK
PUT NEW TAPE ON TWO, THEN PRESS START
USED MORE THAN FIVE CARDS FOR ONE SPECIFICATION
INVALID CHARACTER IN A NUMBER
FIRST LETTER NOT A C, M, E, OR S
INVALID SUBSCRIPT
TWO SUBSCRIPTS ARE EQUAL IN OBJECTIVE FUNCTION
NO +, -, GE, LE, =, OR E AFTER AN X(I)
A COMMA DOES NOT TERMINATE THE END OF A CONSTRAINT OR THE OBJECTIVE FUNCTION
ERRORS NOT CHECKED IN CARD NUMBER XXXX

COLUMN 80 IS NOT A BLANK
CONSTRAINT NUMBER IS GREATER THAN 21
A COMMA IS BEFORE THE END OF A SPECIFICATION

C. TYPED WITH EACH ERROR STATEMENT EXCEPT THE FIRST TWO IS

ERROR IS IN CARD NUMBER XXXX

(ALL CARDS ARE COUNTED.)

V. EXAMPLE INPUTS

```
$$JOB STAFF  CASSIDY.HENRY
$$ALPS
MAXIMIXE  = .5X(1) + 6X(2) + 5X(3)  .
SUBJECT TO THE FOLLOWING CONSTRAINTS.
CONSTRAINT (1) 4X(1) + 6X(2) + 3X(3)  LE 24  .
CONSTRAINT (2)  X(1) + 1.5X(2) + 3 X(3) LE 12   .
CONSTRAINT (3)  3X(1) + X(2) LE 12  .
S AN EDITORIAL STATEMENT
END PROBLEM
```

```
$$JOB FAC    ABLE. B.
$$ALPS
MAXIMIZE IT  ONE TIME = 2E00X(1) +3 X (2).
    S AN EDITORIAL STATEMENT
CONSTRAINT (5)-X(2) GE -3 .
CONSTRAINT (4) X(1) LE 3 .
CONSTRAINT (3) X(1) +5X(2) GE4.
CONSTRAINT (2) 6E00 X(1) +2E00X(2) GE +8.
CONSTRAINT (1) X(1) + X(2) LE4 .
END
```

```
$$JOB STAFF SMITH. HARRY
$$ALPS
MINIMIZE THE FUNCTION = 5.30X(1)+4.90X(2)+4.40X(3)+5.10X(4)+7.00X(5)+6X(6)
       +5.70X(7)+5.20X(8)+5.80X(9)+6.70X(10)+7.00X(11)+8.00X(12)+6.30X(13)
        +5.90X(14)+5.40X(15)+6.10X(16).
SUBJECT TO THE FOLLOWING
CONSTRAINT(1) 1X(1)+1X(2)+1X(3)+1X(4)LE13.
CONSTRAINT(2) 1X(5)+1X(6)+1X(7)+1X(8)LE10.
CONSTRAINT(3) 1X(9)+1X(10)+1X(11)+1X(12)LE8.
CONSTRAINT(4) 1X(13)+1X(14)+1X(15)+1X(16)LE4.
CONSTRAINT(5) 1X(1)+1X(5)+1X(9)+1X(13)=10.3.
CONSTRAINT(6) 1X(2)+1X(6)+1X(10)+1X(14)=7.1.
CONSTRAINT(7) 1X(3)+1X(7)+1X(11)+1X(15)=6.2.
S   EDITORIAL STATEMENT--AS IF I HAD ANYTHING TO EDITORIALIZE
CONSTRAINT(8) 1X(4)+1X(8)+1X(12)+1X(16)=9.1.
END OF PROBLEM
```

VI. EXAMPLE OUTPUT

```
MINIMIZE THE FUNCTION = 5.30X(1)+4.90X(2)+4.40X(3)+5.10X(4)+7.00X(5)+6X(6)
       +5.70X(7)+5.20X(8)+5.80X(9)+6.70X(10)+7.00X(11)+8.00X(12)+6.30X(13)
        +5.90X(14)+5.40X(15)+6.10X(16).
SUBJECT TO THE FOLLOWING
CONSTRAINT(1) 1X(1)+1X(2)+1X(3)+1X(4)LE13.
CONSTRAINT(2) 1X(5)+1X(6)+1X(7)+1X(8)LE10.
CONSTRAINT(3) 1X(9)+1X(10)+1X(11)+1X(12)LE8.
CONSTRAINT(4) 1X(13)+1X(14)+1X(15)+1X(16)LE4.
CONSTRAINT(5) 1X(1)+1X(5)+1X(9)+1X(13)=10.3.
CONSTRAINT(6) 1X(2)+1X(6)+1X(10)+1X(14)=7.1.
CONSTRAINT(7) 1X(3)+1X(7)+1X(11)+1X(15)=6.2.
S   EDITORIAL STATEMENT--AS IF I HAD ANYTHING TO EDITORIALIZE
CONSTRAINT(8) 1X(4)+1X(8)+1X(12)+1X(16)=9.1.
END OF PROBLEM
```

MATRIX CHECK

	1	2	3	4	5	6	7	8
1	1.0000	1.0000	1.0000	1.0000	0.0000	0.0000	0.0000	0.0000
2	0.0000	0.0000	0.0000	0.0000	1.0000	1.0000	1.0000	1.0000
3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
4	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
5	1.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000
6	0.0000	1.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000
7	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000	1.0000	0.0000
8	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000	1.0000

	9	10	11	12	13	14	15	16
1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
3	1.0000	1.0000	1.0000	1.0000	0.0000	0.0000	0.0000	0.0000
4	0.0000	0.0000	0.0000	0.0000	1.0000	1.0000	1.0000	1.0000
5	1.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000
6	0.0000	1.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000
7	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000	1.0000	0.0000
8	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000	1.0000

	17	18	19	20	21	22	23	24
1	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
3	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000
4	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000
5	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000
6	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000
7	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000
8	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000

	25
1	13.0000
2	10.0000
3	8.0000
4	4.0000
5	10.3000
6	7.1000
7	6.2000
8	9.1000

CONTRIBUTION (PROFIT OR COST)

	1	2	3	4	5	6	7	8
1	-5.3000	-4.9000	-4.4000	-5.1000	-7.0000	-6.0000	-5.7000	-5.2000
	9	10	11	12	13	14	15	16
1	-5.8000	-6.7000	-7.0000	-8.0000	-6.3000	-5.9000	-5.4000	-6.1000
	17	18	19	20	21	22	23	24
1	0.0000	0.0000	0.0000	0.0000	-500000.0000	-500000.0000	-500000.0000	-500000.0000

BASIS VARIABLES

1	2	3	4	5	6	7	8
17	18	19	20	21	22	23	24

COSTS

	1	2	3	4	5	6	7	8
1	0.0000	0.0000	0.0000	0.0000	-500000.0000	-500000.0000	-500000.0000	-500000.0000

SOLUTION MATRIX

	1	2	3	4	5	6	7	8
1	0.0000	1.0000	0.0000	1.0000	-1.0000	0.0000	-1.0000	0.0000
2	0.0000	0.0000	0.0000	-1.0000	1.0000	1.0000	1.0000	0.0000
3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
4	0.0000	0.0000	0.0000	1.0000	-1.0000	-1.0000	-1.0000	0.0000
5	1.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000
6	0.0000	0.0000	0.0000	-1.0000	1.0000	1.0000	1.0000	0.0000
7	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000	1.0000	0.0000
8	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000	1.0000

	9	10	11	12	13	14	15	16
1	0.0000	1.0000	0.0000	1.0000	-1.0000	0.0000	-1.0000	0.0000
2	0.0000	0.0000	0.0000	-1.0000	0.0000	0.0000	0.0000	-1.0000
3	1.0000	1.0000	1.0000	1.0000	0.0000	0.0000	0.0000	0.0000
4	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000	1.0000
5	0.0000	-1.0000	-1.0000	-1.0000	1.0000	0.0000	0.0000	0.0000
6	0.0000	0.0000	0.0000	-1.0000	1.0000	1.0000	1.0000	0.0000
7	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000	1.0000	0.0000
8	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000	1.0000

	17	18	19	20	21	22	23	24
1	1.0000	0.0000	1.0000	0.0000	-1.0000	0.0000	-1.0000	0.0000
2	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	-1.0000
3	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000
4	1.0000	0.0000	1.0000	1.0000	-1.0000	-1.0000	-1.0000	0.0000
5	0.0000	0.0000	-1.0000	0.0000	1.0000	0.0000	0.0000	0.0000
6	-1.0000	0.0000	-1.0000	0.0000	1.0000	1.0000	1.0000	0.0000
7	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000
8	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000

	25
1	4.5000
2	.9000
3	8.0000
4	1.4000
5	2.3000
6	2.6000
7	6.2000
8	9.1000

47

OPPORTUNTIY COSTS OR SHADOW PRICES

	1	2	3	4	5	6	7	8
1	0.0000	0.0000	0.0000	.9000	.7000	.1000	.3000	0.0000
	9	10	11	12	13	14	15	16
1	0.0000	1.3000	2.1000	3.3000	0.0000	0.0000	0.0000	.9000
	17	18	19	20	21	22	23	24
1	1.0000	0.0000	.5000	0.0000	499993.7000	499994.1000	499994.6000	499994.8000

VARIABLE USED	QUANTITY	CONTRIB./UNIT	VALUE
2	4.5000	-4.9000	-22.0500
18	.9000	0.0000	0.0000
9	8.0000	-5.8000	-46.4000
20	1.4000	0.0000	0.0000
1	2.3000	-5.3000	-12.1900
14	2.6000	-5.9000	-15.3400
3	6.2000	-4.4000	-27.2800
8	9.1000	-5.2000	-47.3200

OPTIMAL FUNCTIONAL VALUE = -170.5800

VII. PROGRAMED BY HENRY CASSIDY, AIDED BY BILL PETEFISH, AUG. 6, 1965

1620 SPS and SPS II-D

Object Deck Modifier

Betty M. Earlougher
Stanford Electronics Laboratories
Stanford University

December 6, 1965

1620 SPS and SPS II-D Object Deck Modifier

I'm sure most of us have had the experience of writing and debugging as SPS program on the 1620. Typically, the process goes as follows:

1. Write and key punch the SPS source program.
2. Arrange for computer time. 2 or 3 hours will be needed for a moderately long program.
3. Load the assembler deck. This takes 2-3 minutes.
4. Load the source program and execute Pass I of the assembly
5. Load the source program again, execute Pass II of the assembly, and get an object deck punched.
6. Load the object deck into core.
7. Find a bug or bugs. These may occur in object deck patches, if any have been made in the deck.
8. If the change necessary to correct the bug is minor, punch a patch card to correct it. Also, if large portions of the program were not destroyed by the error, patch in core and continue debugging. Otherwise, it may be necessary to reload the object deck, or even to re-assemble.

With a disk file and Monitor, the process is made somewhat easier by bypassing the tedious loading of the assembler. Even so, debugging is a frustrating procedure, and not the least of these frustrations can be the errors made in correcting the errors in the original program.

When a bug is found, a correction can often be made in the object deck, avoiding the assembly process. This is done by punching a patch card, which is loaded over the erroneous instructions or data when the object deck

is loaded. When patching a program, the address at which the correction is being made and the correction itself usually must be typed twice--once to enter the correction into core and once to punch a patch card. Errors can be made in a number of places in this process--

1. Typing the address at which the change is to be made.
2. Typing the change.
3. Punching the change in a patch card.
4. Punching the address in a patch card.
5. In the card assembler, computing and punching the address at which the change ends.
6. Keeping track of which change goes with which patch card.

This catalogue of errors may sound exaggerated, but after 3 or 4 hours on the computer, it is possible to make any or all of these errors in any given correction.

A few years ago, a student in the Electrical Engineering Department of Stanford University wrote a program which prevents the person debugging an SPS program from making four of the six errors mentioned above. This program is called the 1620 SPS Object Deck Modifier, or DECK MØD for short. It was originally written to be operated with the card assembler, and was revised when SPS II-D under Monitor I became available.

Let me describe the program with reference to the SPS II card assembly system; modifications necessary for other versions of SPS are minor.

DECK MØD is loaded after the object program is in core. During loading, a message is typed giving the two sense switch functions used. DECK MØD requires 1000 digits of storage, and usually resides in the upper 1000 digits of core. The first instruction is then at 19000, 39000, or 59000, depending

on the core size available, so one can easily remember the starting address.

To use DECK MØD, execute a branch to the starting address. The typewriter carriage will return and the program will wait for an entry from the typewriter. The address at which the modification is to be made should be typed. This address is the lowest memory address (high order digit of the modification). It need not be a 5-digit number--the program will supply leading zeroes. When R/S is depressed, the typewriter will space over and wait for entry of the actual modification. Up to 61 digits of numeric information may be entered; record marks are allowed only as the last digit. When R/S is again depressed, a 3-digit sequence number is typed to identify the change and a patch card is punched in the proper format with the sequence number just typed. After the patch card is punched, the patch is transmitted as a record to the appropriate place in memory. The sequence number will not be reset to 000 until DECK MØD is reloaded--thus there will be no duplicate sequence numbers in a series of modifications. As long as the patch cards are kept in order, there is no difficulty in making patches on top of patches.

Two program switches are used by DECK MØD. Switch 4 is for the typical error-correction procedure. (Turn the switch on after a typing error is made, hit R/S, and the information may be re-entered.)

Switch 3 is used to provide a return to the program being debugged. It may be turned on whenever the typewriter is awaiting data entry (either an address or a modification). The words "GO TO" will be typed, and the typewriter will await entry of an address to which a branch is to be made. (Again, this need not be a 5-digit number.) When R/S is depressed, a branch to the address just typed will be executed. The patch cards are usually allowed to accumulate in the punch hopper until the object program is to be

reloaded, at which time they are placed in front of the seventh card from the end in the object deck.

Operation of DECK MØD is essentially the same for SPS II-D (disk SPS). A modification may have 67 digits, with no record marks except as the last character. All patch cards are punched in absolute format, so DECK MØD may not be used to patch relocatable SPS (e.g. subroutines written to be used with a FØRTRAN main program). Patch cards are placed in front of the second card from the end of the object deck. The deck is self-loading, rather than stored on the disk, to avoid the possibility of destroying a part of core by calling the Monitor. Of course, it would load faster if stored on the disk in core-image format, and this can certainly be done if memory below 2402 is never used by the programmer.

In summary, the 1620 SPS Object Deck Modifier provides a way to avoid the most common errors associated with correcting errors in a program, by automatically punching patch cards at the same time a correction is made to a program in memory.

This program will be available from the 1620 Program Library in the near future. Until then, I will be happy to supply copies of the source decks.

ABSTRACT

CLEARTRAN is a system for compiling FORTRAN statements to yield an object program having maximum efficiency. The object program generally occupies less than one-half the core space and usually executes twice as fast as the MONITOR II system. Programs involving substantial amounts of subscripted variables may execute in as little as one tenth the usual time.

"Infinite" programs may be compiled by virtue of "instant" linkage from disk and the use of optional advanced language concepts.

"My 1620 can draw circles around your 1620" aptly describes the Format capability. Equations can be "plotted" on the printer. Information can be extracted from a card read or a card may be re-read by any Format number. Complete printer control is available with FORMAT statements. Printing of the results of one problem may be obtained while computing the next set of answers.

Error analyses are exceedingly thorough at both the compile and execution stage. For example, unidentified variables, and out-of-range subscripts are called out at both compile and execute time. The object program seldom blows up during execution. A tract routine is available for presentation of both the name of the variable and its value as calculated.

CLEARTRAN VERBS

ACCEPT
ASSIGN
BRANCH BACK
CALL
- EXIT
- INTERRUPT
- LINK
- PDUMP

COMMON
CONTINUE
DEFINE
- ADDRESSES
- DISK
- DISK ADDRESS
- FAST LOG
- SIZE
- START

DIMENSION
DO
DO BACK
END
EQUIVALENCE
FETCH
FIND
FORMAT
FUNCTION
GO TO
HOLD
- CARD IMAGE
- ERROR MESSAGE
- PRINT SKIP
- PRINT ROUNDING

IF
INTEGER
PAGE
PAUSE
PERFORM
PRINT
PUNCH
READ
REAL
RECORD
RELEASE
- CARD IMAGE
- ERROR MESSAGE
- PRINT SKIP
- PRINT ROUNDING

RETURN
ROUTINE
SET
STACK
STOP
STORE
- ADDRESSES
- CONSTANTS
- NAMES

SUBROUTINE
TAG
TRACE
- PRINT, TYPE, PUNCH
- OFF

TYPE
ZIP

IN-LINE ROUTINE

A group of FORTRAN statements prefaced by ROUTINE is defined as an in-line routine. The routine is given a name with up to six alphanumeric characteristics, the first of which must be alphabetic. A routine is normally entered by means of the PERFORM command and the normal exit is by BRANCH BACK.

The PERFORM command generates a BTM (branch and transmit immediate) type of instruction, with the return address being carried to the routine for use when a BRANCH BACK instruction is encountered. If the PERFORM command includes a statement number, the BRANCH BACK will be to the address of the statement number specified; otherwise, the return address will be that of the statement following PERFORM.

The data and variables used in a routine are identical to those of a mainline program. A routine may be located anywhere in the program except within the confines of another routine. The normal exit from a routine is by the BRANCH BACK command, of which several may be used if desired. A direct entry to any numbered statement of the routine may be used. In this event, the BRANCH BACK exit address from the routine will be that specified by the PERFORM command last used to enter the routine.

The address of a routine may be stored in a subscripted array by the STORE ADDRESSES command. This makes it possible to PERFORM a computed address, e.g., PERFORM RUTEN(J).

While within a routine, one may PERFORM other routines provided the chain of addresses required to return to the mainline program is not broken. If there is a need to break the chain, the address of the ROUTINE where the break is to occur may be saved by including the name of the routine as the third operand of the PERFORM command.

By this means, one may perform a ROUTINE from within the routine itself, if desired. Examples are illustrated in the sample programs.

DEFINITION OF VARIABLES AND ASSIGNMENT OF ADDRESSES

A variable is "defined" if it is encountered to the left of an equal sign, in a READ, ACCEPT or FETCH statement, or in one of the following: COMMON, DIMENSION, EQUIVALENCE, INTEGER, REAL, STACK.

DIMENSION

The DIMENSION statement is used to define the size and the number of words in an array. The number of subscripts which may be used is not limited to three. The length of the fields in an array can be made different from normal by placing an intergal number in front of each element of the list. The minimum length field is two. There is no maximum limit to the length of the field; however, the practical limit for use in conjunction with printer commands is 288.

COMMON

This command is identical to IBM's.

STACK

The STACK command is the opposite of COMMON; i.e., the variables in the list are assigned sequentially ascending addresses, while those in COMMON have descending addresses. A dimensioned variable can be equivalenced to the first element of a list previously stacked making it possible, thereby, to refer to the list as an array. The STACK command is especially convenient for use in conjunction with the deferred PRINT command described later.

TAG

If it is desired to determine all the positions in a program where reference to a particular variable is made, the TAG command may be used. Up to five variables may be tagged at one time; e.g.,

TAG, V, VOICE, A20, I, YOU

REAL

This verb defines a variable as a floating point type even though the initial letter might be I, J, K, L, M or N.

INTEGER

This command defines a variable as a fixed point type even if the initial letter is other than I, J, K, L, M or N.

FIELD LENGTH

As with the DIMENSION command, the length of a variable can be changed from the normal length by placing a number immediately after the STACK, REAL or INTEGER commands.

ALTERNATE DIMENSIONING

The list in the COMMON, STACK, REAL or INTEGER commands can be subscripted as in a DIMENSION statement if the variable is to be dimensioned. (The variable must not then appear in a DIMENSION statement.)

STORE

The STORE command may be used to store at compile time three types of data: addresses, fixed or floating constants with sign, or alphanumeric data. The name of the field where the data are to be stored in core is the first element of the list; the remaining elements are stored in first and successively higher addresses.

ASSIGN

This command may be used to move addresses in core at object time.

DEFINE

The DEFINE ADDRESSES command may be used to specify indirect addresses where the actual address of a subroutine, routine, statement number, constant, etc., may be found. This command is very useful for a communication link between two programs which may be in core simultaneously.

TRACE

The commands TRACE PRINT, TRACE TYPE, TRACE PUNCH, and TRACE OFF may be used to follow the path of a problem through a program. The TRACE PRINT statement calls in a relocatable subroutine which prints the name of the variable on the lefthand side of the equal sign in each arithmetic statement, together with its numerical value. Three variables and their values are printed on each line. Switch 4 activates the TRACE subroutine at object time.

CALL INTERRUPT

The CALL INTERRUPT statement results in storage on disk of the core immage of the program and data, together with the address at which the interrupt occurred. The CALL INTERRUPT command may be selected

by program calculations or by turning on a programmed sense switch. The program may be restored and execution continued by loading a single INTERRUPT RESTART card. If data remain to be read, the last card read, together with cards not yet read, should be set aside for reloading. Obviously, data stored by RECORD commands may be lost unless the disk pack is set aside.

PRINTER DUMP

If it is desired to dump the contents of core on the printer during the execution of a program, one may use the following command:

CALL PDUMP (N1, N2)

N1 and N2 may be absolute addresses or they may be the names of variables. CALL PDUMP pulls in a disk utility program which prints core between the limits specified, with 100 digits per line grouped by tens, with core addresses conveniently shown. Control returns to the program after execution.

ZIP

The ZIP statement used immediately preceding a DO will speed up the evaluation of arithmetic expressions involving subscripted variables. The conditions where ZIP may be used are:

1. Indexing must be under DO loop control only.
2. Each arithmetic statement must be complete without having to use more than one continuation card.
3. DO's may be nested not more than ~~four~~ three deep.
4. The number of subscripted words may not exceed 20.

(The above limits are tentative.)

Example:

```
      ZIP
11    DO 3 J=1, 10
12    DO 3 K=3, 5
13    DO 3 L=1, JIM, 2
 1    A(J) = A(J) + B(J, K, L) +AB(I)
      IF (C(J,K,L) -50. )2, 3, 2
 2    C(J,K,L) = D(J,K+1, L-1) +F +A(J)
 3    CONTINUE
```

Note there are four subscripted "words" in the above, under ZIP control. A(J) is one word, B(J, K, L), C(J, K, L) and D(J, K+1, L-1) are the others. Each word must consist of not more than 12 characters including the two parentheses. AB(I) is not a "word" because its index, I, is not under DO control.

Another ZIP could be used after statement No. 3. The DO's must have numerical starts and numerical increments. The DO's may have variable upper limits; i.e., JIM in statement 13.

I/O FEATURES

The input or output achieved by execution of conventional I/O FORTRAN statements is identical to that from IBM compilers. Certain additional features are available, as follows:

1. B-TYPE, G-TYPE, and J-TYPE FORMAT
2. ALPHABETIC SUBSTITUTION IN E & F OUTPUT
3. AUTOMATIC E & F ROUNDING
4. AUTOMATIC PAGE SKIP
5. CARD IMAGE REREAD
6. DEFERRED PRINTING
7. FORMAT OVERRIDE
8. NON FORMAT READ
9. PLOTTER SIMULATION
10. PLUS SIGN (+) SPACE IGNORE
11. COMPLETE PRINTER CONTROL

B-TYPE FORMAT

B-type words (B for Beta) can be read or written with a non-standard length designated by the FORMAT statement. The length may be a single character, B1, or up to 80 characters for reading the entire card, B80, or B144 for printing 144 characters. The variable where such a word is stored normally is dimensioned so as to have its word length equivalent to that used in the FORMAT. For example B1 words could be read into a dimensioned variable with a word length of at least 2. B80 words should be read into fields which are at least 160 digits long.

A single B-type character is stored as two digits in core, the left one of which is flagged. If stored in a standard fixed-point field of four digits, only the two positions to the right side of the four are used. When printing or punching such a word, by a B-type FORMAT which is identical in length to that used to read the word, conventional output is obtained. However, if the length designated by the FORMAT statement used to read the word is less than the length used in output, the character will be positioned incorrectly by the difference in the two lengths. This can be corrected by changing the output FORMAT statement so that the two lengths are equal, using X-type FORMAT to make up the differences where required for spacing purposes.

G-TYPE FORMAT

This is identical to F-type FORMAT except that the first blank following the field is considered as a decimal. G-type FORMAT is normally used for reading only.

J-TYPE FORMAT

This is identical to I-type FORMAT except that the first blank to the right of the field terminates reading. This permits left-justified, fixed-point fields.

ALPHABETIC SUBSTITUTION IN E & F OUTPUT

On occasions it may be desirable to substitute a short word or blanks in place of an E or F output field. This can be done by setting the floating-point variable to be listed equal to a special alphabetic field. The two leftmost characters of the alphabetic field must be the decimal point equivalent (03). The decimal point is not printed but the alphabetic characters which represent the remaining portion of the word will be printed. As an illustration,

```
    IF (A) 2, 1, 2
  1 A = WORD (1)
  2 PRINT 100, A
100 FORMAT (F10. 0)
    STORE NAMES (WORD(1), .←NONE, .     , .ALL)
```

will cause the word NONE to be printed if A were zero at the IF statement. Blanks would be printed if A were set equal to WORD(2) and the word ALL would be printed if set equal to WORD(3).

AUTOMATIC E & F ROUNDING

Rounding of E and F output is automatic. If automatic rounding is not desired, it can be bypassed by the command: HOLD ROUNDING; and restored later if desired by: RELEASE ROUNDING.

AUTOMATIC PAGE SKIP

A skip to a new page is automatic when the bottom line of the page is sensed (printer indicator 34). This feature can be eliminated by the command: HOLD SKIP (and restored by RELEASE SKIP). This might be desirable when using plotter simulation, or when page skip is under program control.

CARD IMAGE REREAD

The usual READ statement with a FORMAT number will cause a new card to be read and data extracted therefrom in accordance with the FORMAT specifications. It is possible to "read" the same card again, using different FORMAT statements if desired, by two different methods:

1. Place an X (for extra) after the FORMAT number in the READ statement.
2. Use the command, HOLD CARD IMAGE, followed by a conventional READ statement.

The latter causes a one-time skip of the normal procedure whereby a new card is read for each READ statement, making it possible to reread the last read card. The HOLD CARD IMAGE command can be nullified by the command RELEASE CARD IMAGE.

DEFERRED PRINTING

Some problems require that all or almost all of the calculations be completed prior to doing any output. With CLEARTRAN it is possible to print the results of one set of calculations while calculating the following set. By this means, printing and calculations can go on simultaneously with a considerable savings in time. Deferred output can be obtained by placing the letter "S" (for Save) after the FORMAT number of an output statement; e.g., PRINT 102S, List. This command will be ignored at object time prior to execution of a SAVE command.

The output commands utilizing this feature can be placed at selected positions in the mainline of the program where recycling does not occur. Alternately, all PRINT S commands can be placed in a ROUTINE using a computed GO TO to execute successive statements. The ROUTINE could be executed by randomly placed PERFORM statements. When used in a ROUTINE, the "printer busy" indicator should be tested to save time (if the printer is busy an immediate exit from the ROUTINE should be made.)

The SAVE (V1, V2) commands result in a transfer of that portion of core image lying between the address of variable V1 and variable V2 to a safe place in memory. This includes V1, but not V2. The variables to be listed by S type output commands should be in contiguous memory locations for the least space requirements. The STACK or COMMON commands are used to achieve the desired order. All the variables to be listed should be in either COMMON or STACK, but not part in one and part in the other.

The deferred output command can be made to list current values if a zero is used in a SAVE statement; i.e., SAVE (0). The original status can be restored by using a negative number in a SAVE statement; e.g., SAVE (-1).

FORMAT OVERRIDE

The list of an I/O statement is under control of the specifications set up in the FORMAT statement. This normally requires that the number of items in a subscripted list be identical to the "repeat" number of the corresponding element in the FORMAT statement.

With CLEARTRAN, an I/O list may involve subscripts using a variable index. The corresponding "repeat" number of the FORMAT specification should be greater than the maximum possible value of the index(99 is tops). If the repeat number happens to be less than the index variable, FORMAT control will pass to the next element of the FORMAT statement.

NON FORMAT READ

The preparation of input data to be read under FORMAT control requires extra care to insure proper positioning of data on the punched card. This problem can be circumvented by using the READ statement without a FORMAT number. Data of the E, F, I and A type can be read without a FORMAT number. One or several spaces are used to separate data fields. All 80 columns of a card may be used. A relocatable library subroutine examines the input data and discriminates between E, F, I or A data. The F-type conversion results from a decimal point in a numeric data field. The E-type results if a decimal point and the letter E are found. The I-type is generated when there is no decimal point in a numeric field. The A-type is obtained if none of the above conditions are met.

An "input error" is called out if the variable being read is in the wrong mode. The unread portion of the card is typed and a BRANCH TO the program starting address occurs.

One card may contain information which is read by several READ statements. After all the fields on a card are read, the next item on a list will cause a new card to be read, even if the item is in the middle of a list. A record mark in column one of a card calls EXIT.

PLOTTER SIMULATION

The SET command is identical to the PRINT command with the exception that printing does not occur. The SET command is normally used to build an image which is to be held in position for additional modifications. If an X follows the FORMAT number of a SET command, "extra" information can be placed in the image without destruction of information previously placed (except that which is overlaid). By this means, it is possible to build up a complex line of information which is to be printed after all the information is in place.

The X specification in a FORMAT statement is used to position or space adjacent fields. In CLEARTRAN one may use the X specification followed by a fixed point variable in parenthesis; e.g., X(N1). This specification will result in a number of spaces equal to the value of the fixed point variable N1.

One may use the space supress character (+) in column one to print one set of characters on top of another. This character should be erased (1H0) prior to the final PRINT command.

After a line of data is in position, it is printed by a PRINT command, the FORMAT number of which is followed by the letter X.

The first 80 characters of an image can be punched by placing the letter X after the FORMAT number of punch statement.

A program to illustrate these features is attached. This program plots three equations, coordinate grids, and prints alphabetic information simultaneously. Another program "draws" a picture of a heat exchanger tubesheet.

PLUS SIGN (+) SPACE IGNORE

It is not necessary to provide space for the plus sign when printing or punching. This makes it possible to put additional information on a card when punching, or to pack E or F fields adjacent to other fields when printing. An error may occur if the E or F fields are negative, since space must be provided for the minus sign.

COMPLETE PRINTER CONTROL

A complete set of printer controls is available with CLEARTRAN. The following is a list of the printer controls which are achieved by placing a Hollirith character in column one:

Before Printing		After Printing	
+	Space supress		
J	one space	S	one space
K	two spaces	T	two spaces
L	three spaces		
1	skip to channel 1	A	skip to channel 1
2	skip to channel 2	B	skip to channel 2
3	skip to channel 3	C	skip to channel 3
4	skip to channel 4	D	skip to channel 4
5	skip to channel 5	E	skip to channel 5
6	skip to channel 6	F	skip to channel 6
7	skip to channel 7	G	skip to channel 7
8	skip to channel 8	H	skip to channel 8
9	skip to channel 9	I	skip to channel 9
=	skip to channel 11	.	skip to channel 11
@	skip to channel 12	)	skip to channel 12

(Any other character may result in a run-away carriage.)

FORMAT statements with multiple slashes which are executed only by PRINT commands are automatically compiled so as to take advantage of fast printer spacing insofar as possible.

The "printer busy" indicator (35) can be sensed by use of the statement: IF (SENSE SWITCH 35) N1, N2. A BRANCH TO statement N1 occurs if the indicator is on (buffer is unavailable for loading); N2 if off (buffer can be loaded).

Similarly, 33, 34 and 25 can be used to sense respectively channel 9, channel 12 , and printer check indicator on the 1443.

A U T O M O N I T O R

F O R T H E

I B M 1 6 2 0

Presented At The

WESTERN REGION WINTER

MEETING OF COMMON

December 7, 1965

By

John W. Rettenmayer

Western Data Processing Center

405 Hilgard Ave.

Los Angeles, California 90024

The WDPC 1620 Automonitor was written at the Western Data Processing Center, which is a part of the Graduate School of Business Administration at UCLA. It was written to serve as a debugging aid to the students in BA 113, an introductory course in data processing. This class usually has from 60 to 80 students, so any debugging which requires manual operation at the console is extremely inconvenient to other students and greatly increases the confusion from too many students milling around awaiting their turn on the machine. (The 1620 lab is an open-shop arrangement.)

The WDPC Automonitor is used for debugging student-written 1620 machine language programs. Instead of being executed directly, the student's program is executed one instruction at a time, with each instruction and its memory address being printed out just before it is executed. As long as the Automonitor is in control, the tracing may be turned on or off to enable the operator to trace selected portions of the student's program, or all of it. For example, tracing may be desired only after a certain point in the program has been reached, that point being indicated by a particular value being printed. At that point switch 1 may be turned on and the instructions will be traced from then until switch 1 is turned off again. This selection process may be repeated as often as needed.

The students are given only two constraints: (1) their programs must start at location 5000, and (2) they may not use any memory positions lower than 5000 because that area is reserved for the loader program and the Automonitor. To increase the throughput of the system, the students' machine language programs are run under the Monitor I System. This requires that the student program return control to Monitor upon successful execution. If execution

is not completed successfully, then the operator must manually branch to Monitor (using a 4900796 instruction). In order to allow stacking of jobs, the student cannot test for last card, as is usually done to signal the processing of the last data card. Therefore, each student is also required to test for a trailer card having a record mark in column 1. The Monitor I end-of-job card does nicely for this purpose.

As implied above, the 1620 configuration at WDPC includes a 1311 disk storage drive and a Monitor I System. We also have 40K of core storage, and, although it has no bearing upon the use of the Automonitor, a 1627 plotter.

O P E R A T I N G P R O C E D U R E

Student's deck setup:

≠≠COLD START

≠≠PAUSE (for stacked input)

≠≠JOB

≠≠XEQSMACHLG

(Student's machine language program,
punched one instruction per card.)

(blank card -- to separate program from data)

(data for student's program)

≠≠≠≠

If tracing will be desired during any part of the user's program execution, switch 2 should be turned on before the word EXECUTION is typed on the console typewriter. Then turning switch 1 on at any time will cause tracing to begin, and turning switch 1 off will stop the trace printout.

LOADER PROGRAM

MACHLG (a listing of which follows later) is the program which loads the student's program. MACHLG is stored on the disk in the regular way, with an associated DIM entry, and is called into core and executed by the ≠≠XEQS Monitor Control Card. The functions performed by MACHLG are the following.

First, it loads from the disk into core 37,000 digits, starting at location 3000 and ending with 39999. These 37,000 digits are obtained from one cylinder of the disk, which contains 20,000 digits, so that part of the cylinder is used twice. (We have redefined the disk storage so that cylinder 0 is not a part of the disc's working storage, but is available for MACHLG to use. Other installations may wish to obtain the cylinder in some other way, and should change the instructions in MACHLG accordingly.) The first 14 sectors of the cylinder contain the Automonitor program; therefore it is loaded with its starting address being 3000. The rest of the cylinder contains only zeroes which are used to effectively clear core from the end of the Automonitor to location 39999. Locations 0 - 2999 are not cleared since they contain the arithmetic tables, part of the Monitor Supervisor routine, and MACHLG itself.

Second, MACHLG loads the student's program, starting at location 5000. MACHLG loads each instruction, i.e. the first 12 digits of each card, into successively higher memory positions until it detects a card with a blank in column 1, signifying that the entire program has been loaded.

Third, control is transferred to the instruction at 3000, which is the first instruction of the Automonitor.

AUTOMONITOR OPERATION

The following description is brief and perhaps confusing; please refer to the program listing for better understanding.

The Automonitor first interrogates Switch 2 to see if tracing is or will be desired. If not, control is transferred to the student's program and that program executed directly. If Switch 2 is on, then the student's program will not be executed directly, but will be simulated by the Automonitor in the following manner.

First, the Automonitor copies the student's first instruction into an area within the Automonitor program, which will be referred to as the simulation register. (Actually the instruction is copied into two areas -- one for execution and the other for outputting the instruction.) If the instruction is not a branch instruction, it will be executed directly in the simulation register. Since this register in imbedded in the Automonitor program, control will remain in the Automonitor after execution of the instruction. Then the next sequential instruction of the student's program is treated by the same process.

If the instruction is a branch, then the actual execution of that instruction, even in the simulation register, would cause control to be transferred to the student's program and the Automonitor would then be inoperable. Therefore, that branch instruction must be simulated in such a way that the next instruction operated upon by the Automonitor is the correct one. That is, if a branch were called for by the conditions of the machine, then the instruction at the branch address must be the next one fetched and treated. If a branch were not called for, then we want to take the next

sequential instruction after the unsuccessful branch instruction. This is done by carrying out the appropriate tests, for conditional branches, and modifying the 'next instruction address' (ADDR) accordingly.

TRACING

If at any time the user wants to have his program traced as it is simulated by the Automonitor, he only has to turn on switch 1. This will cause the address of each instruction and the instruction itself to be printed. If switch 3 is on, the trace will be punched on cards; if it is off, the trace will be printed on the console typewriter. Which option is chosen depends, of course, on the demand for operating time and the availability and convenience of listing equipment. (The punched output will have the instruction address and the instruction separated by one zero instead of blanks since alphameric mode is not used.)

If it is definitely known that a trace will not be needed, then switch 2 should be off initially so that the user's program will be executed directly instead of being simulated. However, on short student programs the added time used by the Automonitor is insignificant, so one need not worry about accidentally leaving switch 2 on.

RESTRICTIONS

By their nature, operation codes 07, 17 and 27 cannot, to my knowledge, by simulated since to do so would require accessing the hardware registers. Also, the Automonitor will not handle any compare operation, since the Automonitor itself makes a great many comparisons and, as it now stands, would probably destroy

the indicator set by the user's compare operation before his branch-on-condition instruction would be simulated. It is possible to rewrite the Automonitor to handle compares, but the compare instruction is rarely used by beginning students, so we have not done so. For the same reason of infrequent use the branch-no-indicator instruction (op code 47) has not been included either, although to do so would simply be a matter of following the logic of OP46 (see listing of Automonitor).

The WDPC Automonitor could be elaborated upon to a considerable extent to achieve a sophisticated tool for debugging 1620 machine language programs. However, it was the opinion of the author that such a tool would serve to defeat the purpose for which the student is taught machine language. That purpose is primarily to acquaint him with the basic level operations of the computer, and much of that acquaintance comes from manual debugging, i.e. experience. However, a minimal automonitor is helpful in those cases where the lab assistant must help the student debug -- formerly by manually stepping through the program at the console.

Each section of the Automonitor program delineated by the lines of asterisks is self-contained and has no particular physical relationship to the other sections. The 'B FLAG' instruction just before the section labelled 'FLAG' seems to be a superfluous instruction, but it was left in the code in order to preserve this modularity. Extensions to the Automonitor can be easily made, if desired, by following the pattern of the present sections. It is recommended that the modularity be retained since it clarifies the flow of the program to a considerable degree.

ACKNOWLEDGEMENTS

MACHLG was originally written and implemented by George M. Schoenherr, who was in charge of the 1620 installation at WDPC and is now an employee of IBM at San Jose. Sally Ann and Susan Gulick did a considerable amount of debugging of the Automonitor after the first rough version was written. Their work was done as an extracurricular project for BA 113, and is much appreciated.

The author and the WDPC staff would appreciate any comments and would particularly like to receive information on extensions and modifications that are made to the Automonitor.

```
‡‡JOB 5

‡‡          BOOZER, G.L.  BA 113 PROGRAMMING PROBLEM NO. 1

‡‡XEQSMACHLG
EXECUTION

 02150
 00350
 99200
 100200
 110000
 70000
 02005
 01995
END OF JOB
```

The above execution of the student's program was with switch 2 off. It will now be run again with both switch 1 and 2 on, but switch 1 will be turned off after the second answer is obtained.

```
‡‡JOB 5

‡‡          BOOZER, G.L.  BA 113 PROGRAMMING PROBLEM NO. 1

‡‡XEQSMACHLG
EXECUTION

 05000  361500100500
 05012  450503615001
 05036  321500100000
 05048  261206015005
 05060  361500100500
 05072  321500100000
 05084  261206515005
 05096  321206600000
 05108  221207012070
 05120  321207100000
 05132  221207512075
 05144  211207012060
 05156  211207012065
 05168  460532401400
 05180  251500600400
 05192  261500512070
 05204  340000000102

 05216  381500100100 02150
  00350
  99200
  100200
  110000
  70000
  02005
  01995CONDITION CODE NOT RECOGNIZED
END OF JOB
```

```
‡‡JOB 5

‡‡SPS 5

*LIST TYPEWRITER
*ID NUMBER0842
*NAMEMACHLG
*STORE CORE IMAGE

       START  SK    FIELD,00701                    02402 34 02606 00701
              RDN   FIELD,00702                    02414 36 02606 00702
              SK    VELD,00701                     02426 34 02620 00701
              RDN   VELD,00702                     02438 36 02620 00702
       CARD   RNCD  IMAGE                          02450 36 02639 00500
              SF    IMAGE                          02462 32 02639 00000
              CM    IMAGE + 1,00,10                02474 14 02640 00000
              CF    IMAGE                          02486 33 02639 00000
              BE    3000                           02498 46 03000 01200
       TRANS  TD    PROG,IMAGE,67                  02510 25 02638 02639
              AM    TRANS + 11,01,7                02522 11 02521 00001
              AM    PROG,01,7                      02534 11 02638 00001
              CM    TRANS + 11,IMAGE + 12,7        02546 14 02521 02651
              BE    RESTOR                         02558 46 02582 01200
              B     TRANS                          02570 49 02510 00000
       RESTOR TFM   TRANS + 11,IMAGE,7             02582 16 02521 02639
              B     CARD                           02594 49 02450 00000
       FIELD  DDA   ,1,00000,200,03000             02606 00006 100000
                                                   02612 00003 200
                                                   02615 00005 03000
       VELD   DDA   ,1,00030,170,23000             02620 00006 100030
                                                   02626 00003 170
                                                   02629 00005 23000
       PROG   DSA   5000                           02638 00005 05000
       IMAGE  DSS   80                             02639 00080
              DEND  START                          02402

END OF ASSEMBLY.
02720 CORE POSITIONS REQUIRED
00022 STATEMENTS PROCESSED


DK LOADED MACHLG 0842 1046000040240202402‡
END OF JOB
```

```
         DORG 3000,,  DEFINE ORIGIN                          03000
FIRST    TFM  ADDR,5000,7, INITIALIZE ADDR                   03000 16 04168 05000
         TFM  ADDR1,5000,7, INITIALIZE ADDR1                 03012 16 04173 05000
START    BC2  CLEAR,,,IF SW 2 IS ON, BRANCH TO CLEAR
                                                             03024 46 03048 00200
         B    5000,,,BRANCH TO 5000                          03036 49 05000 00000
**********
CLEAR    S    COUNT,COUNT,, CLEAR COUNTER                    03048 22 04189 04189
TD       TD   INST,ADDR,211,TRANSMIT INSTRUCTION DIGIT
                                                             03060 25 05175 04168
         TD   INST1,ADDR,211,TRANSMIT INSTRUCTION DIGIT
                                                             03072 25 05128 04168
         AM   TD+6,1,7,ADJUST INSTUCTION ADDRESS             03084 11 03066 00001
         AM   TD+18,1,7,ADJUST INSTRUCTION ADDRESS
                                                             03096 11 03078 00001
         AM   ADDR,1,7,INCREMENT ADDR                        03108 11 04168 00001
         AM   COUNT,1,10, ADD 1 TO COUNTER                   03120 11 04189 00001
         C    TWELVE,COUNT,, TEST FOR COMPLETION             03132 24 04191 04189
         BH   TD,,,IF P IS GREATER THAN Q, BRANCH TO TD
                                                             03144 46 03060 01100
         TFM  TD+6,INST,7, RESTORE INSTRUCTION ADDRESS
                                                             03156 16 03066 05175
         TFM  TD+18,INST1,7,RESTORE INSTRUCTION ADDRESS
                                                             03168 16 03078 05128
         BNC1 FLAG,,, IF SW 1 IS OFF, BRANCH TO FLAG
                                                             03180 47 03324 00100
         BC3  PUNCH                                          03192 46 03276 00300
PRINT    RCTY ,,, RETURN CARRIAGE ON TYPEWRITER              03204 34 00000 00102
         WNTY ADDR1-4,,, PRINT OUT INSTRUCTION ADDRESS
                                                             03216 38 04169 00100
         SPTY ,,, SPACE ON TYPEWRITER                        03228 34 00000 00101
         SPTY ,,, SPACE ON TYPEWRITER                        03240 34 00000 00101
         WNTY INST,,, PRINT OUT INSTRUCTION                  03252 38 04175 00100
         B    FLAG,,,                                        03264 49 03324 00000
**********
PUNCH    TD   ADDR1+1,RETURN+20,, CLEAR RECORD MARK
                                                             03276 25 04174 04160
         WNCD ADDR1-4                                        03288 38 04169 00400
         TD   ADDR1+1,INST+12,,                              03300 25 04174 04187
         B    FLAG                                           03312 49 03324 00000
**********
FLAG     SF   INST,,, SET FLAG                               03324 32 04175 00000
         CM   INST+1,41,10                                   03336 14 04176 00041
         BNH  EXEC ,,,                                       03348 47 04068 01100
         CM   INST+1,49,10, TEST FOR 49 OPERATOR             03360 14 04176 00049
         BE   OP49                                           03372 46 03540 01200
         CM   INST+1,46,10, TEST FOR 46 OPERATOR             03384 14 04176 00046
         BE   OP46                                           03396 46 03552 01200
         CM   INST+1,45,10, TEST FOR 45 OPERATOR             03408 14 04176 00045
         BE   OP45                                           03420 46 03876 01200
         CM   INST+1,47,10, TEST FOR 47 OPERATOR             03432 14 04176 00047
         BE   OP47                                           03444 46 03900 01200
         CM   INST+1,44,10, TEST FOR 44 OPERATOR             03456 14 04176 00044
         BE   OP44                                           03468 46 03912 01200
```

```
         CM    INST+1,43,10, TEST FOR 43 OPERATOR     03480 14 04176 00043
         BE    OP43                                   03492 46 03936 01200
         RCTY                                         03504 34 00000 00102
         WATY  MESS3                                  03516 39 04311 00100
         B     796                                    03528 49 00796 00000
**********
 OP49    B     BRANCH                                 03540 49 03996 00000
**********
 OP46    SF    INST+8,,, SET FLAG                     03552 32 04183 00000
         CM    INST+9,14,10, TEST FOR OVERFLOW INDICATOR
                                                      03564 14 04184 00014
         BNE   CC9                                    03576 47 03612 01200
         BV    BRANCH,,, BRANCH ON OVERFLOW           03588 46 03996 01400
         B     RETURN                                 03600 49 04140 00000
 CC9     CM    INST+9,9,10, TEST FOR LAST CARD INDICATOR
                                                      03612 14 04184 00009
         BNE   CC1                                    03624 47 03660 01200
         BLC   BRANCH,,, BRANCH LAST CARD             03636 46 03996 00900
         B     RETURN                                 03648 49 04140 00000
 CC1     CM    INST+9,1,10, TEST FOR COND CODE 1      03660 14 04184 00001
         BNE   NOT1                                   03672 47 03708 01200
         BC1   BRANCH                                 03684 46 03996 00100
         B     RETURN                                 03696 49 04140 00000
 NOT1    CM    INST+9,2,10, TEST FOR COND CODE 2      03708 14 04184 00002
         BNE   NOT2                                   03720 47 03756 01200
         BC2   BRANCH                                 03732 46 03996 00200
         B     RETURN                                 03744 49 04140 00000
 NOT2    CM    INST+9,3,10, TEST FOR COND CODE 3      03756 14 04184 00003
         BNE   NOT3                                   03768 47 03804 01200
         BC3   BRANCH                                 03780 46 03996 00300
         B     RETURN                                 03792 49 04140 00000
 NOT3    CM    INST+9,4,10, TEST FOR COND CODE 4      03804 14 04184 00004
         BNE   NOT4                                   03816 47 03852 01200
         BC4   BRANCH                                 03828 46 03996 00400
         B     RETURN                                 03840 49 04140 00000
 NOT4    WATY  MESS1                                  03852 39 04193 00100
         B     796,,, BRANCH TO END OF JOB ROUTINE    03864 49 00796 00000
**********
 OP45    BNR   BRANCH,INST+11,11, BRANCH NO RECORD MARK
                                                      03876 45 03996 04186
         B     RETURN,,,                              03888 49 04140 00000
**********
 OP47    B     NOGOOD                                 03900 49 03960 00000
**********
 OP44    BNF   BRANCH,INST+1,11                       03912 44 03996 04176
         B     RETURN                                 03924 49 04140 00000
**********
 OP43    BD    BRANCH,INST+1,11                       03936 43 03996 04176
         B     RETURN                                 03948 49 04140 00000
**********
 NOGOOD  RCTY                                         03960 34 00000 00102
         WATY  MESS2,,,                               03972 39 04253 00100
         B     796,,,                                 03984 49 00796 00000
**********
```

```
BRANCH  SF    INST+2,,, SET FLAG                          03996 32 04177 00000
        BNF   NOFLG,INST+6,, TEST FOR INDIRECT ADDRESS
                                                          04008 44 04044 04181
        TF    ADDR,INST+6,11, TRANSMIT FIELD              04020 26 04168 04181
        B     RETURN,,, BRANCH TO RETURN                  04032 49 04140 00000
NOFLG   TF    ADDR,INST+6,, TRANSMIT FIELD                04044 26 04168 04181
        B     RETURN,,, BRANCH TO RETURN                  04056 49 04140 00000
**********
EXEC    CM    INST+1,38,10, TEST FOR OUTPUT INST          04068 14 04176 00038
        BE    SPACE                                       04080 46 04116 01200
        CM    INST+1,39,10, TEST FOR OUTPUT INST          04092 14 04176 00039
        BNE   INST1                                       04104 47 04128 01200
SPACE   SPTY  ,,, SPACE ON TYPEWRITER                     04116 34 00000 00101
INST1   DSS   12,,, DEFINE FIELD (INST1)                  04128 00012
RETURN  TF    ADDR1,ADDR,, TRANMIT FIELD                  04140 26 04173 04168
        B     CLEAR,,, BRANCH TO CLEAR                    04152 49 03048 00000
**********
ADDR    DC    5,5000,, INITIALIZE ADDR TO 5000            04168 00005 05000
ADDR1   DC    5,5000,, INITIALIZE ADDR1 TO 5000           04173 00005 05000
        DC    1,@,,  SET RECORD MARK                      04174 00001 ‡
INST    DSS   12,,, DEFINE FIELD (INST)                   04175 00012
        DC    1,@,,, SET RECORD MARK                      04187 00001 ‡
COUNT   DC    2,0,, INITIALIZE COUNTER TO 0               04189 00002 00
TWELVE  DC    2,12,, SET VALUE OF TWELVE AT 12            04191 00002 12
MESS1   DAC   30,CONDITION CODE NOT RECOGNIZED@,,         04193 00060
MESS2   DAC   29,OP CODE CANNOT BE PROCESSED@,,           04253 00058
MESS3   DAC   39,OP CODE GREATER THAN 41 NOT RECOGNIZED@,,
                                                          04311 00078
**********
        DEND  FIRST                                       03000
```

ANTHROPOLOGY AND THE TEACHING OF PROGRAMMING

Presented at Western Region Winter Meeting of COMMON, December 7, 1965, Los Angeles, California.

This discussion, since what I have to say could not be considered a "paper" will be mainly narrative in nature.

Let me introduce myself as neither a programming expert nor as an anthropologist. The title of this talk came about due to an unimaginative attempt to call it something. I am, in fact, a physicist in charge of a computational group, trapped into spending part of my time as an administrator. I am involved in a small amount of research in atomic and molecular spectra and structure. I teach a course combining quantum mechanics, atomic and molecular physics.

Usually when a course is given we concern ourselves with the winners--those who pass--and consider the losers as somehow deficient and incapable. But I believe we must look more carefully at the losers in programming courses. Somehow the "miracles" of computing have been greatly overdramatized, and we are all guilty. How else do we get money from tight fisted administrations? Well and good, a lot of people believe things of this sort. Especially students--especially some of the foreign students.

The consideration of attitudes came up in the careful examination of the types of students who were dropping Mathematics 195, Introduction of Automatic Digital Computing--and their reasons.

Mathematics 195 is required of all engineering students at Illinois, and may be taken after having had differential calculus. It became obvious that foreign students were having considerable difficulty, much more than native born, although there had not been much distinction between these two groups in their mathematics grades. More than once, a section would lose every foreign student; only occasionally would a foreign student last through to complete the course.

As far as native born students are concerned, the ability to pass the course seemed to be, in very general terms, only a function of intelligence as evidenced by grades. Good students do well, poor students do poorly. Now and then an otherwise average scholar does very well because something seems to click. However, the foreign student who has been doing average, or perhaps even better than average, work in his other courses, would be in trouble in the programming course. What and why?

For several reasons this is not a trivial problem. Let us look at some aspects of possible answers, which hopefully might lead to better ways of dealing with the problem.

1. Are we overtraining foreign students? Are the skills and experience he is acquiring not applicable when he returns to his native country? Does the student know this and is this why he is negligent in his learning? Some people are becoming analytical and critical, and think that this may be the case. Have we alienated these graduates from the manpower and techniques of his own country, replacing them with skills and tools that are not available? There is no point in saying what ought to be available; we must deal with what is. Are we right in demanding a skill that cannot be applied?

The facts are that there are now 80,000 foreign students in this country, most of whom are studying in scientific and engineering fields. If we require

that they learn useless techniques, upon returning to their native lands they become estranged from their own people. In such an instance, unfortunately education has become a destructive influence. This need not and should not be.

2. Let us consider our system of enumeration. Is it the only "useful" system? Well, let us look at Roman engineering: roads, forts, arches, the Coliseum, and aqueducts that still function. All done with an awkward I, II, III, IV, etc., and look at this $\sqrt[IV]{\text{LXXXI}} = \text{III}$ Try that on your 1620, 1800 or 360. We are decimalized almost everywhere in the world, but should we be, to coin an expression, binarized or octalized? I would like to learn more about learning arithmetic. Because most of us learned by counting objects, apples and bananas, long before we learned to abstract "numberness". In some societies this "numberness" is prone to have associated ideas which I shall touch on later.

3. Is there a cultural and ethnic antipathy to our objectives? Let me quote:

> -non-literate peoples are capable, under the pressing conditions of necessity, of doing their utmost with their minds to solve some practical problem in a scientific manner. Most of the scientific processes involved are of a practical nature, and there is little time or inclination for science for science's sake. The latter activity does not appear until the development of highly sophisticated societies like Hellenic Greece. This is but yesterday in the history of human time. Because the Greeks were an aristocratic society based on slavery, they, who had so much of the necessary theoretical knowledge at their disposal, virtually failed to apply it. Machines were unnecessary since slaves could do all the work. The Greeks were interested in ideas--in brains--not in drains. The refuse of civilization could be disposed of by slaves, but only those with the necessary leisure could create and maintain that civilization. The Greeks developed the greatest ideas in the humanities and the sciences that the world has ever known, and probably the fewest inventions of a mechanical kind. Not that

> the Greeks were uninterested in the practical application of some of their ideas, it is simply that they were not enamored of the possibility of creating machines that think and human beings who don't.[1]

Are we "thinking" when we "solve" a problem using a machine? I say the best use of machines depends upon our ingenuity as human beings solving human problems. Do some foreign cultures induce inhibitions so that the machine would deprive people of the ability and right to make human judgments?

4. Has the computer technology and use given rise to a form of "patois" or jargon, difficult but possible for the native born, but impossible for the foreigner? We have gone to the stage of requiring more academic work in English from foreigners than from native born. The trouble seems to originate from penalizing a student in his rhetoric class and then penalizing him for not using the ungrammatical syntax of compiler languages. Apparently, if he is hit from two sides for errors that seem to contradict each other, a very frustrating situation is created. But the problem goes beyond just language and words. The people in the U. S. seem to be impressed by large quantities and numbers and except for 13, of which the superstition is vanishing, all numbers are very much the same, but for foreigners numbers have a semantic meaning over the merely technical context. Hindus avoid prime numbers; among the Japanese numbers mean good lock, wealth, bankruptcy and even death. This incidently spills over into the telephone systems. Good numbers command a high price, while unlucky numbers as assigned to the unwitting foreigners.

5. Is our philosophy of classroom behavior ineffective for teaching and learning when used with foreign students? Do they require a more

---

[1]Ashley Montagu, "Man: His First Million Years," World Publishing Co., 1957.

personal, more intimate, interpersonal relationship? Are we lacking in cross cultural communication? Does this lack affect our ability to educate our own native born students? I want to give you a few ideas to mill over. My source for some of this material is a paper bound book "The Silent Language" by Edward T. Hall--"how people talk to each other without words."

Assume, if you will that what is ordinary courteous behavior in the U. S. does not offend a foreigner (example: among Hindus one must not sit with soles of feet toward another person--it showed great disrespect; among Arabs one must not pass things on with the left hand.)

Are we making a mistake in believing that people should "understand" what they are doing?

During World War II, when many technicians were needed it was thought that good mechanical aptitude would be a basic need for good airplane mechanics. To everyone's surprise a good shoe clerk would turn out better than a fellow who had fixed his own car. The important point was not mechanical aptitude but the ability to follow instructions. Someone who "understands" is apt to have his own ideas, which may be just the thing not needed.

In the U. S. we go on a "first come, first serve basis" again for some foreigners this is a peculiar notion. The rank of the person--and people sent to the U. S., England, France, and Germany to study are persons in their own country of high rank--determines when he shall be served. So that the rationals of logical ranking or ordering, as we use it, is a new and disturbing method.

The last concept I wish to mention is time. In the U. S. we want "fast" and faster machines to do more and more. A better machine is a faster machine. This may not be the foreigner's concept. Again the matter

of position in his own society may imply some notions uncomfortable to us but natural to him. The purpose of the computing machine is not to save time but to save labor; he does not care how long something takes to be performed--up to a point--since labor is likely to be considered menial, the entire process of programming and computing may be thought of as an inferior activity.

Let me iterate, 80,000 students in our educational system may be a small percentage, but since these are the technical people we may have to deal with, it pays us, in the broadest sense, to learn to communicate--and at all possible levels.

The 1620 for Simulation in a Biomedical Environment

I. R. Neilsen and J. J. Horning
Scientific Computation Facility
Loma Linda University
Loma Linda, California

INTRODUCTION:

Our 1620 installation handles a job-mix of great variety which would probably seem quite normal to those of you working with a general-purpose machine in a university environment. Batch processing methods are, providing the turn-around time can be kept reasonable, generally satisfactory for much of the work that is being done. We are not going to discuss the handling of such a general job-mix. We will instead concentrate on a particular class of problems which is occupying an increasingly large fraction of our effort and for which the batch processing methods of a conventional computing center are inappropriate and ineffective. These are problems of modeling and simulation of living systems. We would like to summarize for you some of the important hardware and software requirements for successful work of this kind and describe the way in which these requirements are being met on our 1620 (a Mod II with 40K, 1622 Mod II, two 1311's and a CalComp plotter). In order to give specificity to our remarks we will briefly introduce some of our currently active work on three models.

SYSTEM REQUIREMENTS AND DESCRIPTION

Our experience to date leads us to believe that the mathematical model of a biological system will most often consist of a set of differential and/or algebraic equations. These equations will typically be non-linear and the set of equations may turn out to be very large. A first and extremely important system requirement is convenient, effective man-machine communication. This man-machine interaction has been achieved very effectively in the use of analog computers. There are those who feel that the digital computer can never effectively compete in this domain. The digital computer does, however, offer certain specific advantages which make it tempting to try to use it in the study of such models.

In any case, an on-line mode of operation appears absolutely essential to us. The user needs to be able to halt computation, start over again, modify various parameters in his equations and even change the structure of the model in a quick and convenient way. These changes should be accomplished in a time comparable to that required for the model builder to think about them-which is to say that they need to be accomplished in time on the order of seconds. Program changes in milliseconds are obviously not needed and changes requiring minutes or hours to implement are too slow and laborious. Putting all possible values of all parameters into the program in advance in

order that the output from a batch-processing run shall include the behavior of the model in the as-yet-unknown region of interest, is totally unrealistic.

We have made a strenuous effort to perserve the open-shop concept. We justify this in terms not of increased computations/dollar, but in terms of increased utility to our users. We resist the trend towards measuring efficiency in terms of the machine, rather than the man. We fear that model-building and simulation may become tedious to the point of not being done, if users must extrude their models through a closed-shop, batch processing system. Our ideal is to have the researcher who conceived and programmed the model sitting at the console, adjusting parameters and evaluating results.

A requirement which is particularly pertinent in our university situation in which this work is being done on a part-time basis by people who are also teaching classes and doing other things, and in which there tend to be many interruptions between sessions at the computer, has to do with the convenience of getting the model back on the machine. For small problems on analog computers this can frequently be handled by means of plug-in patch boards, but for larger problems this becomes a significant difficulty.

A third system requirement, implied in part by what we have already said, is for on-line graphical output. As the parameters in the model are manipulated, one needs to be able to quickly visualize and evaluate the effect. This is not conveniently done through the medium of pages of tabular output, but rather from the graphical display of the behavior of important system variables. Since the installation of our CalComp 565 graph plotter (alias IBM 1627) we have had a marked jump in computer usage of several types including simulation.

Graphical output permits convenient comparison of model with experiment, and often greatly enhances model comprehension. This is important in both the construction of the model, and in its later use as a conceptual and educational tool.

However, desirable as this plotter is, there are drawbacks which prevent it from being the complete solution to all our problems. For one thing, the 3 inches/sec plotting speed means that on-line plotting can easily consume large share of the available computer time. Second, and somewhat related to this, is the fact that graphs cannot be conveniently viewed while being prepared. This becomes crucial when one wishes to maximize man-machine interaction.

Today's popular solution to this type of problem is the cathode-ray tube display. A glance at the systems typically available, however, reveals that the information displayed needs to be refreshed at a 30 to 60 Hz rate. Very simple calculations show that the 1620 could occupy itself full-time in keeping the CRT decorated, with no time left over for useful computation. Buying a buffer memory to provide scope refresh looked economically unattractive. Add to this

the cost of alphanumeric generators, vector generators, etc., and perhaps you realize why we had nearly abandoned the CRT idea as "nice--but out of our price bracket." Observation of a storage oscilloscope in use as an output device on Dr. Homer Warner's 1620 in Salt Lake suggested a practical and inexpensive (less than $3,000) solution to our problem. Basically the idea was to abandon the continuous refreshing of the CRT display and instead, think of the CRT face as being equivalent to plotter paper; once written on, it stays written. Use the same incremental plotting techniques as does the plotter.

What is involved (see Figure I) is an interface between the 1620 paper tape channel and the plotter which is essentially standard. In parallel with the plotter are two bi-directional counters which accumulate the increments of plotter motion. These counters are fed into standard digital-to-analog coverters to generate CRT deflection voltages. A Tektronix storage oscilloscope does the rest. Due to a built-in "memory grid", a spot once written on the CRT remains there until erased.

The CRT display has proved to be a real boon to our installation. Not only does it allow graphical output to run computation-limited (our theoretical output of 66Kilopoints/second is simply beyond our ability to compute points) but the operator can conveniently monitor output as it occurs, and stop a run--to modify parameters, etc.--as soon as it becomes apparent that something has gone wrong. This frequently saves many long, useless runs on the computer.

The above could probably be said of any on-line CRT. There are some specific advantages of our implementation (besides its low cost), which we feel are worth mentioning: A) The computer sees the CRT as identical to the plotter, thus the operator can select, at will, by turning a switch, among the options CRT, CRT + plotter, and plotter, with no program recognition of this fact being necessary. B) NO reprogramming is necessary to use the CRT. Existing IBM routines such as PLOT and CHAR work unchanged (a faster algothrithm has been programmed at our installation, however, to take advantage of the increased speed available with the CRT). C) When output on the plotter is necessary, as it frequently is--for higher resolution, a permanent record, or multicolor work--its progress can be simultaneously monitored on the CRT.

Two additions have been found desirable in the input end of the man/machine interface. First, some extra sense switches are vital. Second, we have plans for a dozen precision potentiometers. With the installation of analog-to-digital equipment these will be used for the "potentiometer twisting" method of parameter input so familiar to analog computer users. We expect these to aid in more intimate feedback from man to model.

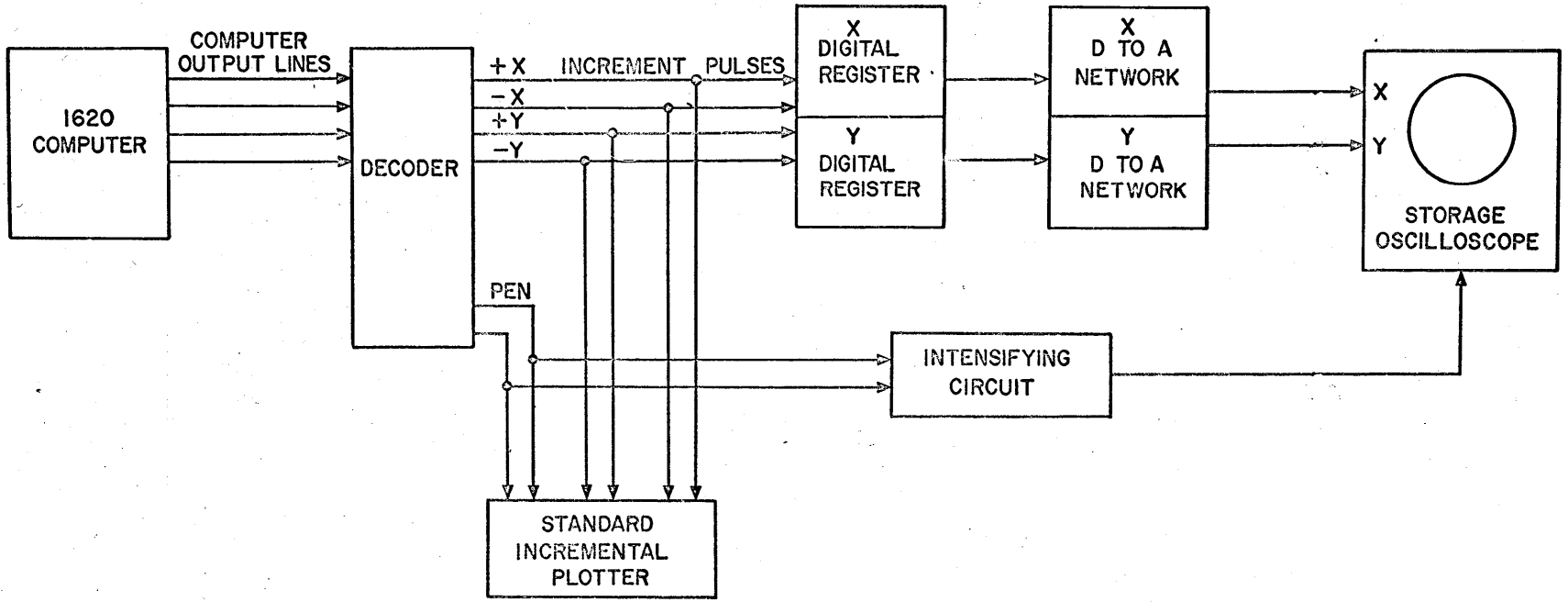

1620
COMPUTER
COMPUTER
OUTPUT LINES
DECODER
+X
−X
+Y
−Y
INCREMENT PULSES
PEN
X
DIGITAL
REGISTER
Y
DIGITAL
REGISTER
X
D TO A
NETWORK
Y
D TO A
NETWORK
X
Y
STORAGE
OSCILLOSCOPE
INTENSIFYING
CIRCUIT
STANDARD
INCREMENTAL
PLOTTER

FIGURE I

Problem solution in real-time has not been an important requirement in our work to date. It has, in fact, been a real advantage to be able to slow down certain physiological processes for the convenience of the observer.

APPLICATIONS

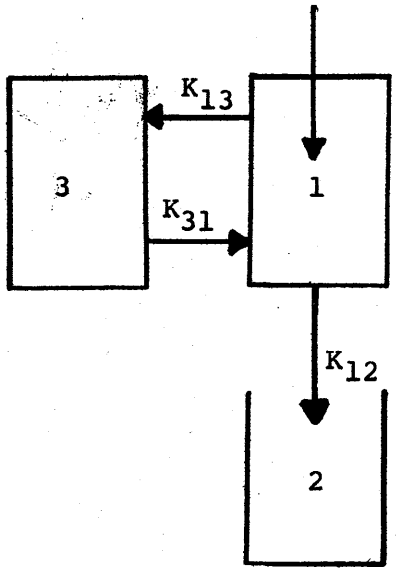
Much of our simulation work involves the PACTOLUS system described by Robert Brennan of IBM, San Jose, at the 1964 Fall Joint Computer Conference. PACTOLUS is one of the so-called "analog-oriented languages." It allows the 1620 to be programmed much like an analog computer: i.e. by the preparation of block diagram and a "wiring list." Many function blocks are available in PACTOLUS. The key element is a numerical integrator, which allows the use of this system to solve differential equations.

Even a digitally-oriented person usually finds it relatively easy to go from a set of differential equations to a block diagram to a PACTOLUS coding sheet with initial conditions, and perhaps 1/2 hour after being given the problem have the problem on the computer. The computer prints on one sheet the system configuration, the initial conditions and parameters, and the time and output controls. Any of these can be selected for modification at any time by turning on various sense switches.

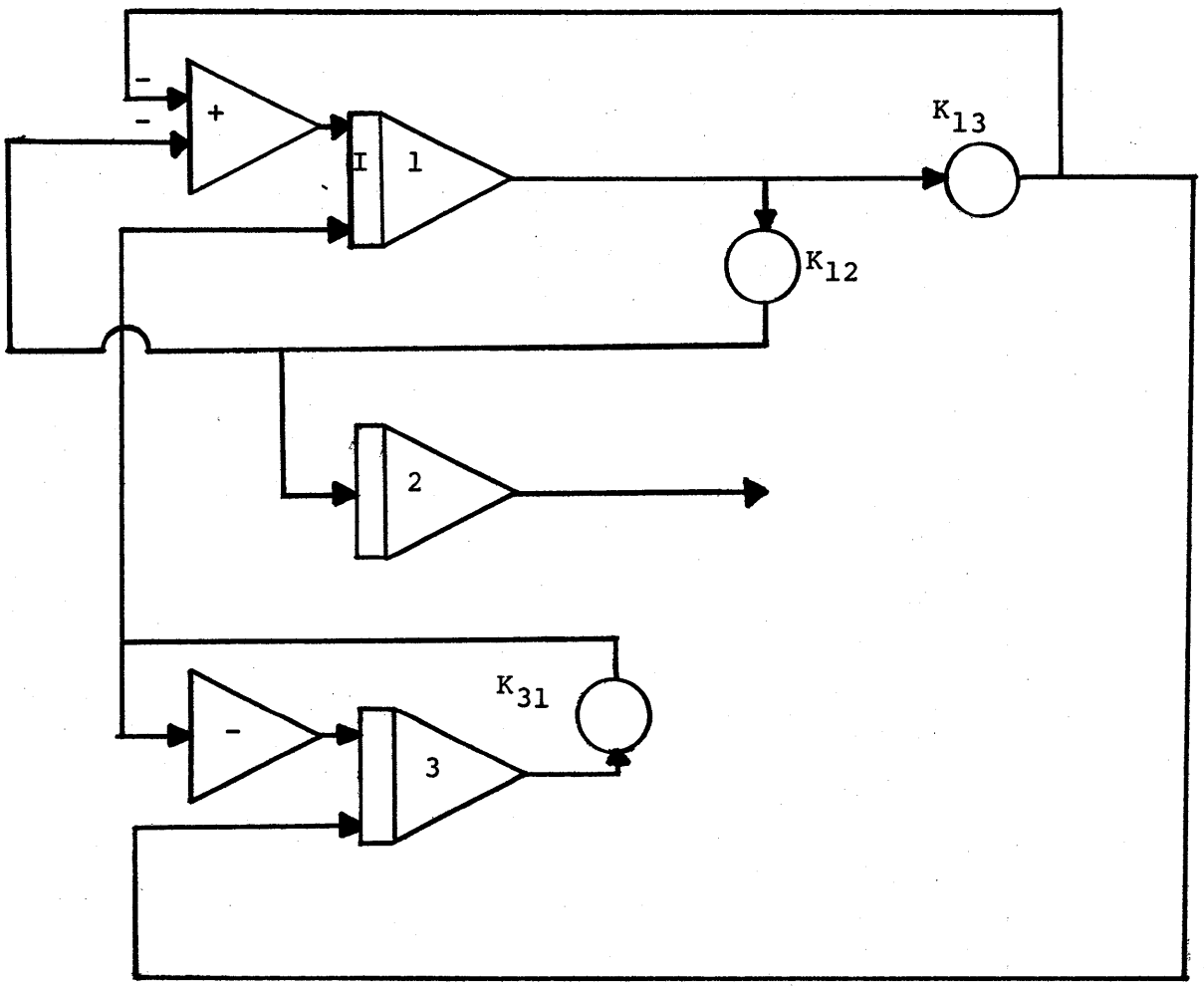
Three models on which we have worked will be used to illustrate typical applications:

KIDNEY MODEL:

The time-course of radioactive tracers injected into the circulatory system can be monitored at various body sites. The radioisotope rhenogram is widely used in evaluating kidney function, but no adequate mathematical model exists for evaluating the curves that are generated by monitoring the radioactivity at kidneys, bladder, and other locations of interest. We are currently working on the development of such a model. A first and rudimentary version began with a so-called two-cavity open compartment system. Such a system can be described by a set of three simultaneous differential equations. (See Figure II) The programming time to get this simple model running on the computer was only a few minutes. Many physiological problems can be similarly considered in terms of such multi-compartment analysis and the ease of programming in PACTOLUS and the ready evaluation of the model through inspection of the solutions in graphical form on the CRT are a great convenience.



TWO-COMPARTMENT KIDNEY MODEL



$$\dot{V}_1 = K_{31} V_3 - K_{13} V_1 - K_{12} V_1$$

$$\dot{V}_2 = K_{12} V_1$$

$$\dot{V}_3 = K_{13} V_1 - K_{31} V_3$$

FIGURE II

NERVE MODEL:

A great variety of structures and processes make up the nervous system. One obviously important and all-pervasive phenomenon is the action potential by which information is transmitted along the membrane separating the solution within from that without. The resting nerve is characterized by ionic concentrations within and without which, operating across the membrane conductance, determine the potential of the solution within the nerve in relation to that outside (typically -80 mV). When stimulated, the nerve "fires" or depolarizes. Actually the potential reverses with the inside going to perhaps +40 mV. This depolarization "spike", or action potential, then propogates along the nerve fiber. Hodgkin and Huxley[1] proposed the model shown in Figure 3 for the situation occuring at the bounding membrane.

Implementing this model is not without challenge. We are not completely satisfied with the present version. (see Figure 4) The model is straightforward (and uninteresting) except for the non-linear conductances shown in series with the ionic "batteries". The values of these conductances have been shown by Hodgkin and Huxley to be complicated functions of the time-history of cell voltage. Without the graphical output and interactive capabilities of our present computer system it is doubtful that a working model on the 1620 would have been achieved at all. Further trial and error adjustment of the functions and parameters is needed to more accurately match the behavior of a real nerve.

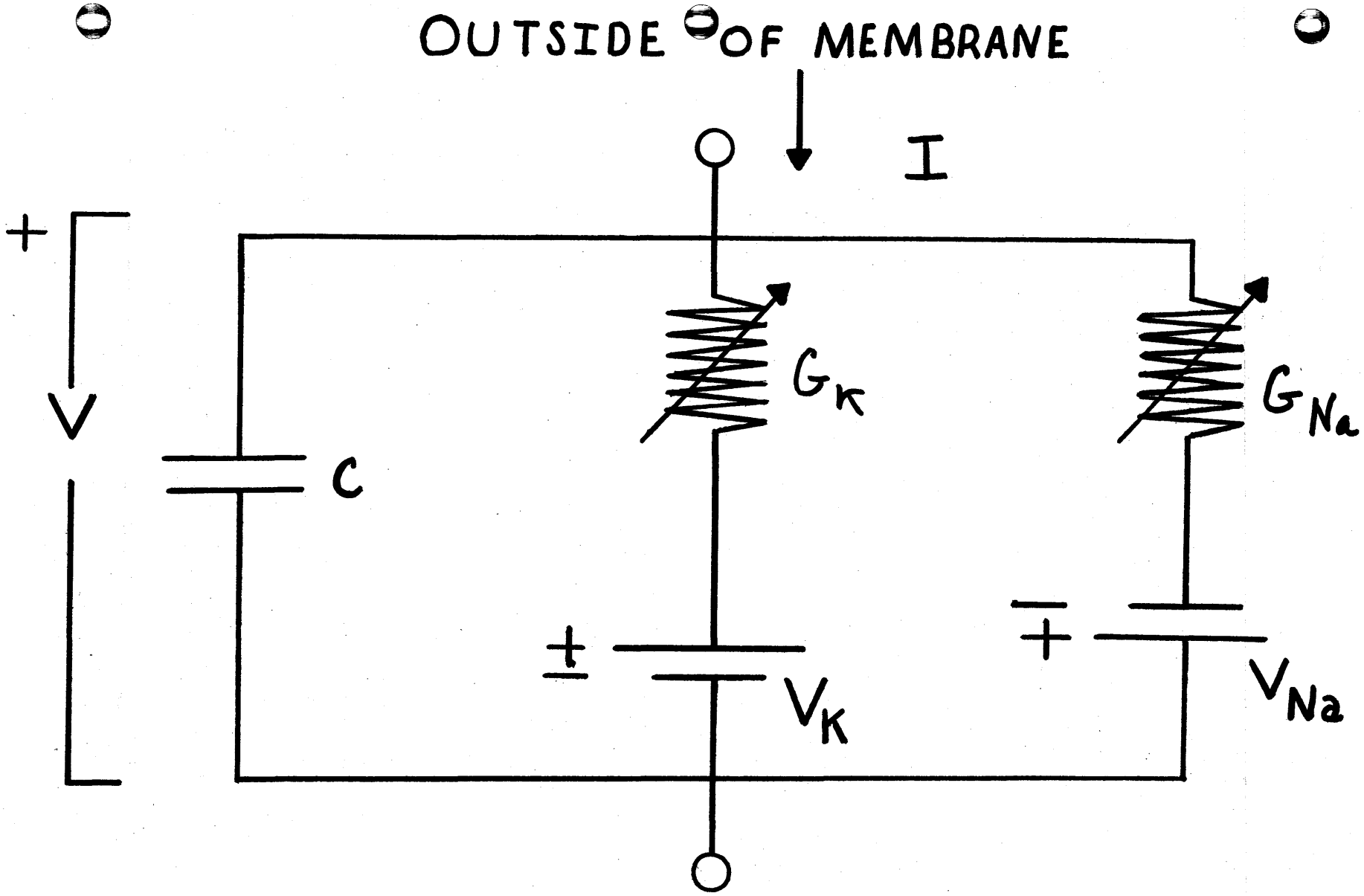
HEART MODEL:

There are two limitations of PACTOLUS that we feel tend to somewhat restrict its usefulness. Digital computer users tend to feel rather keenly the somewhat arbitrary restriction to analog-like 3-input block notation and tend to feel that future software should accept differential equations directly as input--perhaps in a FORTRAN-like format--while perserving the man-machine interactiveness which is the strong point of PACTOLUS.

The second is that a special-purpose system such as PACTOLUS is almost inherently somewhat limited in its capability of producing polished, well-labelled outputs such as are typically needed in the use of already well-developed models.

---

[1]Hodgkin, A. L. and Huxley, A. F.; J. Physiol. 117, p 500 (1952)



92

$$I = C\frac{dV}{dt} + G_K(V - V_K) + G_{NA}(V - V_{Na})$$

$$\frac{dV}{dt} = \frac{1}{C}\left[I - G_K(V - V_K) - G_{Na}(V - V_{Na})\right]$$

FIGURE 3

$$\frac{dV}{dt} = \frac{1}{C} \left[I + G_K (V_K - V) + G_{NA} (V_{NA} - V) \right]$$

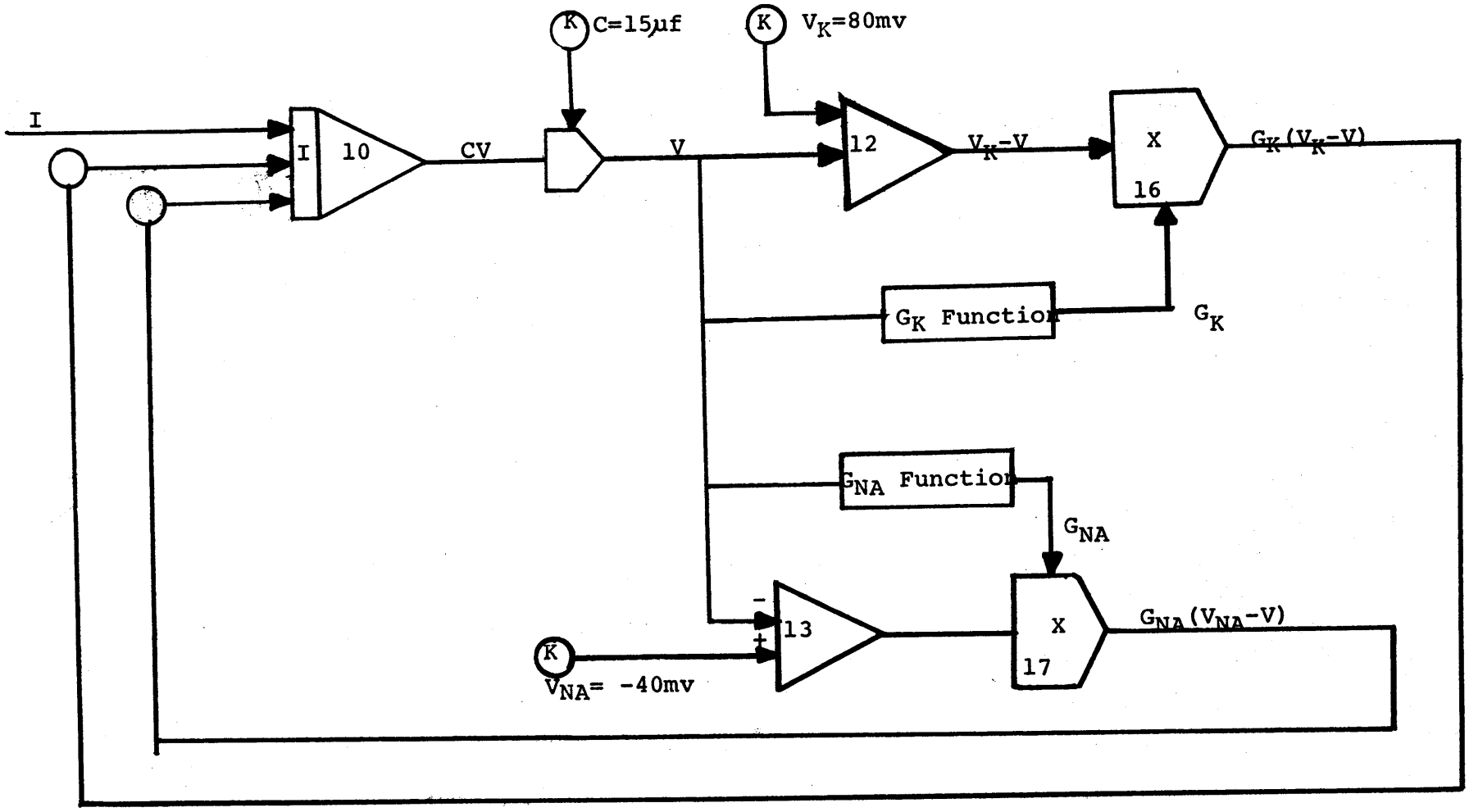


FIGURE 4

The Loma Linda University Heart Model falls into this category. It was a model of proven usefulness as a research tool before conversion to the 1620 was even attempted. Due to the demands of internal logical structure, and output requirements, this model has been written and maintained in the FORTRAN Language.

Computer analysis of electrocardiograms has received much attention. The clinical cardiologist has typically attempted diagnosis on the basis of the so-called scalar electrocardiogram. Leads are attached to the body at various sites and the scalar electrocardiogram is then simply a plot of the voltage between a pair of these leads as a function of time. With the concept of orthogonal leads it becomes possible to think of these voltages as components of an electric vector which changes in magnitude and direction as the depolarization wave sweeps over the heart in a cycle. The figure traced in space by the tip of this vector (or the projection of this figure on a given plane) is then called the vectorcardiogram.

We now have a computer model which will generate vector (or scalar) cardiograms which look like those generated by a human patient.

In developing our computer model we consider a heart divided into 20 segments. As the depolarizion wave sweeps over the heart it passes through a given segment thereby creating an equivalent electric vector. The time course of these twenty vectors is given as twenty functions with their directions specified by their direction cosines. The vector sum of these 20 segment vectors taken at each moment of time then specifies a single equivalent vector which is regarded as generating the vectorcardiogram. The model was first programmed for the analog computer[2] but the 1620 version has been much more successful not only because of greater freedom in specifying the 20 functions but also because of the ability of the digital computer to process the output in useful ways to provide a more meaningful display.

We shall restrict our discussion to techniques by which we have tried to enhance the usefulness of the graphical output of this model.

The typical electro-cardiogram EKG of yesterday and today consists of a polygraph output of from 6 - 12 essentially periodic traces of voltages obtained from leads placed at various points on the body. 12 lines are the basic output of the Heart Model and correspond to one period of each of 12 scalar leads. These can be

---

[2]Selvester, R. H., Collier, C. R. and Pearson, R. B.; Circulation, Vol. xxxl, January 1965.

used to compare the output of the model with traditional EKG's. Two loops, the projections on a horizontal and frontal plane of the 3-dimensional loop, present the same amount of information as the scalar lines, but relationships are much more easily seen.

The use of color permits us to readily compare abnormal cases with the normal ones. Shifts in the vector loops are often detected much more easily than the corresponding changes in the scalar leads.

Our next thought was "if going from 1-dimension to 2-dimensions in our presentation increases the digestibility of information so much, what about 3-d?" We don't have a 3-d plotter yet. What we do have is a series of programs which prepare left-and right-eye views from 3-d coordinates for use with an old-fashioned stereoscope. The 3-d effect so produced can be startingly good.

We are in the process of debugging the programs and the photographic techniques for the preparation of full-color, 3-d, animated motion pictures of vector-cardiograms, using data from either the model or from patients. Right now, the program is ahead of our photographic technique. When both operate to our satisfaction we plan to prepare movies to be used in medical education.

CONCLUSION

We are reminded of a remark by Professor Culler when asked how the time-sharing system at UC, Santa Barbara compares with other such systems. He responded that "one comparison that could be made is that the Santa Barbara system is a member of a class of systems which exist-as opposed to a much larger class of systems which people talk about." Remote terminals and graphical display devices are rapidly becoming available for a variety of computer systems and the on-line concept is one which is supposedly going to revolutionize the world of computers in the near future. Meanwhile our system exists and we are on-line. The economics of the system are such that we can allow significant blocks of user time with the operator interacting with the computer as he observes on the CRT display the effects of his parameter and programming manipulation. The needed element of high-speed CRT display was developed and could be duplicated by other users with a modest expenditure of time and money.

PERT / CPM

PRACTICAL APPLICATIONS AND A LOOK TO THE FUTURE

PRACTICAL APPLICATIONS

I am very pleased for the opportunity to participate in this conference. And, particularly on this subject.

I feel, as do the other members on the panel that techniques like PERT (Program Evaluation and Review Techniques) and C. P. M. (Critical Path Method) have great potential.

Up until the last couple of years, references to PERT/CPM were associated with large defense programs or large construction projects. Now, we find that there is an awakening on the part of management to the potential of these techniques. These techniques (particularly C.P.M.) are now being used in all types of construction (large and small), in the field of education, equipment maintenance, accounting and auditing, data processing, and many others. As a matter of fact, the use of these techniques have spread to so many new fields that I have heard hints to the effect that the notorious "English Train Robbery" was so well planned, scheduled, and carried out that the culprits must have used C.P.M.

Before getting into details on "Practical Applications" let us review various techniques now used in Project Management.

We have PERT, CPM, GANTT CHARTS, CHECK CHARTS, HUNCH, HABIT, INTUITION. If we were to list them in order of use, they would most likely fall in the following sequence:

1. Hunch, Habit, Intuition
2. Gantt Charts
3. Check Charts
4. CPM (Trend toward Activity Oriented systems)
5. PERT (Defense Industry uses, but technique being modified)

You know, businesses are now too complex and costly to operate as they were in the "good old days." And yet, we still find many being run by habit. That is, there is no formal planning.

We continuously hear and read of projects being late for one reason or another. Some part or material not delivered, some equipment or manpower not available, request for budget appropriation overlooked. Hurry and wait, crash program, work overtime, seems to be the standard practice. And, we realize these delays and conditions might have been averted had there been any real planning.

I recently heard of two computer installations that had to be delayed. One was delayed because the site was not ready, the other because systems and programming for the conversion were not started in time.

There will always be delays beyond the control of a manager, but think of the times that the project could have gone smoothly had there been planning; that is: both strategic and operational planning.

I am sure you have heard of the Polaris Submarine and Missile project, and the project being completed two years ahead of schedule using the new PERT technique. Also, of Dupont and others using C.P.M and saving time and money on maintenance of equipment and on construction projects.

But let us discuss some smaller applications of C.P.M. Some that have benefited both the company and the person who developed the network.

Following are some examples of such C.P.M. applications:

I. A young engineer became interested in PERT/CPM and took a course on the technique. His supervisor heard of his interest and assigned him to develop a network for the "Puddingstone Dam Maintenance Project."

Show Chart No. I.

II. A young man working in the construction industry took a course on C.P.M. technique for a term project. He developed a network on a "60,000 sq. ft. Warehouse" that his company planned to build. He showed the network to his superintendent. He was so impressed that the company actually used the network for scheduling and controlling the project.

Show Chart No. II.

III. A Police Sergeant applied the C.P.M. technique to a program for "overhauling and installation of radios on motorcycles." He developed the network and computed the total time manually. It proved the value of formal planning and scheduling. It also pointed out that the project could not possibly be completed by the scheduled (direct date) using the manpower provided.

The Sergeant's superior officers were so impressed with the advantages of this technique over others in use that they are recommending that other officers learn PERT and C.P.M. techniques.

Show Chart No. III.

IV. An Executive Secretary took an interest and learned the fundamentals of PERT and C.P.M. She developed a network on "merging two departments." Her boss, a vice president, was so impressed that he assigned her to develop C.P.M. networks on "opening a European Plant."

Show Chart No. IV.

She now has a new title "Pert Analyst."

There are many applications I could mention: mountain cabins, apartments, painting buildings, D.P. applications and installations, and so forth. You can see that you do not have to look far for practical applications. Any project which meets the criteria: space, people, departments, critical schedules, tight budgets, and interrelated or dependent tasks.

Projects that call for good planning, scheduling and control.

Western Region
Winter Meeting of Common

Gaylord L. Baker

PERT / CPM

PRACTICAL APPLICATIONS AND A LOOK TO THE FUTURE

A LOOK TO THE FUTURE

1. Unfortunately, none of us have a Crystal Ball. Nor do we have the psychic or prognostic ability of Jean Dixon. As a matter of fact, most of us have trouble predicting what our wives or children are going to do next.

   But as far as business is concerned, if we study the past, what's happening right now, we can see trends. It doesn't take a very astute person to evaluate the technological advances made in science and engineering in the last 30 years: our space program, television, surgery on eyes and heart, nuclear power. And of course we must recognize the advances made in computer technology as almost unbelievable.

   What about the Future? We can expect some amazing progress with lazers, with atomic power, electronic miniturization, and in the use of computers.

2. What about management? Have we made much advancement? Certainly not comparable to science and engineering. Oh, we are using computers and some other improved office equipment, but there are so many managers operating by habit; by trial and error.

   They say there is a trend towards the use of mathematics and the computer by management.

3. Future for Management. No question, there will be wider and expanded use of PERT and CPM and, no doubt, with technical improvement. Remember, the primary function of management is PLANNING and PERT andCPM Forces Planning. That is, both strategic and operational.

   STRATEGIC PLANNING:
   - A clear definition of objectives
   - Scope of project defined
   - Directed dates and authority
   - Imposed conditions
   - Laws and legal provisions
   - Approvals
   - Financial and Budget Controls
   - Resource limits and Company Policy

   OPERATIONAL PLANNING
   - What we must do
   - Logical sequence of performance
   - Interrelationships and Dependencies (one task to another)
   - The things we use: manpower, material, equipment, etc.

Another thing, there will be more of a common understanding between those in the financial end (Controllers, Budget Officers, Boards, those controlling the finances) and those doing the project--whether the project would be construction, sales, training, Data Processing, scientific, engineering, or any of the many new fields.

Each will be able to visually see the CONDITION OF THE PROJECT in terms of:

What has to be done
When and how long
What has been done
What is being done
What has yet to be done
Also, <u>cost to date</u>

In closing, a few years ago I was out at UCLA doing a Resource Simulation Study on the 1620. I noticed a young fellow in the corner and assumed that he was the son of one in our GR. He looked about fifteen. He sat down beside me. I asked, "Waiting for dad?" "No, waiting to test PROG." This took me by surprise and I asked, "Prog. 1401?" "No, but 1410." I asked, "how old?" He said, "twelve years."

You know, we live in a competitive society and to advance and to keep ahead of these young students we have to not only have experience, but to keep abreast of new ideas and techniques. Like PERT and CPM.

THE DEPARTMENT OF WATER AND POWER'S CRITICAL PATH METHOD SCHEDULING PROGRAM FOR THE IBM 1620 DATA PROCESSING SYSTEM CADS MK III

(Construction and Design Scheduling Mark III)

Raymond C. Burt, Robert C. Burt and Marciano Lopez
Department of Water and Power
Los Angeles, California

ACKNOWLEDGMENT

The authors wish to acknowledge the assistance given by Mr. L. O. Schmidt, formerly with the Department of Water and Power, Mr. J. T. Au, and Mr. B. H. Kawaguchi, in the programming and analysis of the Department's CADS scheduling system.

INTRODUCTION

This paper is a discussion of the Critical Path Method of scheduling hereafter referred to as CADS now in use by the Department of Water and Power, City of Los Angeles, for the scheduling of design, construction, material procurement, equipment maintenance, and preliminary operations of steam plants, large bulk power substations, Water System projects, and a nuclear plant.

CADS may be described as an automated method of scheduling complex projects on digital computers. More than that, it is a logical approach to project scheduling, organization and planning in detail.

The various methods of computer scheduling (see appendix, page A1) presently available have been developed out of a need for a logical method to analyze the relationship between thousands of activities which go into large manufacturing and construction projects.

WHY IS CADS NEEDED?

CADS plays an important role in the Department's design and construction for several reasons, as follows:

1. In order to supply sufficient detail for the establishment of target dates for such items as material procurement, engineering, preparation of drawings, orderly installation of equipment and so forth, it is necessary to schedule and coordinate several thousand jobs or activities.

2. Since there are over 100 design, construction, and material procurement groups in a major project team, a standard scheduling and coordination technique administered by a central authority is required.

3. Because of the vastness of the projects and the continual need for updating schedules, high speed scheduling techniques are required to prevent delays in obtaining vital information.

4. A method is desired which enables the scheduler or engineer to pinpoint, in advance and with sufficient scope, the areas of difficulty and to take corrective action months in advance instead of after the difficulty has been encountered.

5. A technique is desired which enables a scheduler or engineer to describe a project in regard to its many dependent activities or jobs, with the required start and finish dates of the project, and which then automatically adjusts the resulting schedule by making time corrections on critical jobs to produce a final schedule meeting the original requirements specified.

6. Finally, a technique or method is required which enables the scheduler or engineer to see the effect of a slippage of any one job or activity in a project with regard to the overall project target date.

With regard to the last point above, it has been assumed in the past that there are certain key items or activities in a large project, such as the placement of a main power transformer or the arrival of a turbine spindle, which could be used to measure the progress of a project and indicate project completion date slippage.

Another assumption in the past was that only the large or key items in a project need be scheduled and that everything else would automatically fall in place. Both of these assumptions are false. The two weeks late arrival of reinforcing steel for a transformer foundation can delay a project just as long and can be just as serious a problem as the two weeks late arrival of the transformer to be installed on the foundation.

HOW IS CADS USED IN THE DEPARTMENT?

The Department's version of CPM scheduling is called CADS. CADS, which stands for Construction and Design Scheduling, has been used in the scheduling and the coordination of all design, material procurement, construction and testing activities needed to build three large bulk power substations, Receiving Stations S, P, and T, in the Los Angeles area. Two of these stations, RS-S and RS-T, are conventional, aboveground 138 KV to 34.5 KV Substations with ultimate capacities of 400 MVA each. RS-P is a 230 KV to 34.5 KV underground station, (in the downtown area of Los Angeles) with an initially installed transformer capacity of 320 MVA. All phases of the projects such as site selection, engineering, drafting, management approval, material procurement, construction, testing, and energizing of the stations are scheduled in detail to answer inquiries and problems of the following types:

An electrical design engineer needs to know when a set of detailed structural drawings will be ready in order to know when he can start design of an electrical equipment installation.

A land division officer wonders whether a two weeks delay in obtaining a parcel of land will be acceptable.

A member of engineering management is considering whether or not enough time is left to request a change in system design.

A chief draftsman flooded with a deluge of drafting jobs must assign the order in which the jobs are to be done.

A specification writer is faced with deciding what delivery date will be satisfactory approximately two years from now.

Noting the tremendous expense today for construction equipment rental and wages, a project engineer has the problem of deciding the exact day on which to begin construction in order to meet the project completion time and at the same time minimize cost. These problems, and many more, can be answered by CADS.

A typical receiving station schedule for the type of stations indicated above consists of approximately 2500 jobs or activities. Portions of a typical receiving station schedule are shown in the appendix of this paper. These computer listings will be explained later.

In conjunction with the scheduling of large receiving stations, it was necessary to schedule the transmission and underground engineering and construction of the lines and cables entering the stations.

The other main application of CADS in the Department has been the scheduling of construction and preliminary operation activities of Units 1, 2 and 3 at the Department's new Haynes Steam Plant. These units will produce 230 MW each. The construction of these units not only involves the coordination and scheduling of the Department's own construction groups, but

also that of the various contractors responsible for the boilers, turbines, tank erections, and so forth. To date only the construction and preliminary operation activities have been scheduled by CADS for the above units.

One of the extra dividends accrued in going to computer scheduling, and in particular to CADS, was the increased understanding and knowledge of the detailed activities, with their interrelationships, required in building a plant. The benefits from this one point alone have made the program highly desirable and successful.

The last major point pertains to the manner in which the CPM or CADS automated scheduling program was implemented within the Department.

The computer program was developed at the suggestion of several Department senior engineers concerned with the coordinating and scheduling of large projects. The original programming group consisted of engineers, having previous plant design experience and familiar with computers.

In the latter part of 1960, much time was spent in researching available publications, principally those concerning the Navy's PERT Program. After analyzing the Department's needs and the programs then available, it was decided to write an entirely new program having the input and output features that the Department required.

The engineers, who wrote the computer program,

prepared, with the help of the Station Design Engineers, the first receiving station schedule data. The first steam plant schedule data were prepared by the preliminary operation, construction and scheduling engineers because of their greater familiarity with steam plant construction activities. The task of originating, organizing, operating and implementing computer schedules for all projects is now done by the Department's Design and Construction Division staff engineering schedulers, who have historically been charged with the responsibility for coordinating and scheduling. The original computer programming group now serves in a consulting capacity.

WHAT CAN CADS DO THAT CONVENTIONAL SCHEDULING TECHNIQUES CANNOT?

As was pointed out above, CADS or CPM can analyze thousands of detailed activities, as to their correct time relations. Using conventional bar chart techniques, it is physically impossible and impractical to correctly relate several thousand activities according to calendar dates, to type or print and proof complete schedules for distribution at regular intervals within a satisfactory period of time, much less examine the critical and subcritical paths for areas of possible trouble.

With former conventional techniques, only the major equipment and construction activities could be scheduled, therefore not providing sufficient detail to indicate problem areas. In addition, independent schedules for design,

construction, and material procurement were made by various subsection schedulers in order to aid their engineering groups. This resulted in conflicting schedules, each in different degrees of updating. CADS can update, print, proof, and ready a complete detailed schedule for distribution in a matter of minutes after the need for a change has been recognized.

Lastly, CADS, through its automatic rescheduling features, can adjust a schedule through its ability to find the critical areas or paths, modify certain time estimates, and recast a schedule iteratively until specific project requirements are met.

HOW DOES CADS COMMUNICATE OR PORTRAY ITS INFORMATION TO THE SCHEDULER AND/OR ENGINEER?

The advent of high speed, large storage digital computers, with the ability to output information at the rate of 600 lines per minute and with up to 132 characters on a line, has provided a temptation for programmers to output too much information for humans to digest. To be certain, management, design engineers, clerks, and construction people need detailed scheduling information because they work with details. It is the day-to-day decisions, such as which jobs are to be performed next, how many people are needed on the design staff during the next year, and can part of a group be rotated to new assignments, that plague management. With the ability to provide detailed

information and while still recognizing the frailty of human beings in digesting large amounts of data, the specifications for the CADS output format were devised.

A person reading a schedule is usually concerned with the details which pertain to his group only. For instance, the supervisor of a group of electrical draftsmen is concerned with jobs which require electrical drafting and not with the number of plumbers needed on a particular date. Hence, the schedule output is divided into summaries according to actual Department design and construction groups. A partial list of groups in the Department given individual summary printouts from the CADS schedule may be found in the appendix, page A4. A supervisor is concerned with decisions such as when jobs assigned to his group should be started and finished, how many men should be employed on each, which jobs can be started early, is overtime required, and are the prerequisites to starting a job available (key drawings, engineering calculations, sketches, and so on). A supervisor therefore needs an individual summary for his group which contains the above information.

The two typical approaches can be made when examining a summary schedule. The first type of inquiry might be as follows:

Has a particular job been completed yet?

And the second type:

Which jobs should be started, finished, or active today?

The first type of inquiry requires an alphabetical listing of activities by summary, while the second requires a chronological listing of activities by summary. The CADS output format includes both types of listings for each summary. In addition, it is convenient to have the output format for each summary in both tabular and graphical forms. The alphabetical listing of activities is printed in tabular form and the chronological listing is printed in graphical form (bar chart) for each individual group summary.

In order to discuss the output formats, samples of which are included in the appendix, it is first necessary to define the terms activity, related (prerequisite) activity, critical path, slack time, normal and critical time, and normal and critical manpower.

An activity can be either a task performed or an elapse of time. Examples of tasks performed are engineering studies, drafting, installations, and testing. An example of an elapse of time is the curing of concrete. Before most activities can be started, certain prerequisite activities must be completed first. These prerequisite activities are known as related activities. As an example, the curing of a concrete foundation is the related activity to the activity of placing a transformer on that foundation. To further illustrate the term related activity, the placing of the transformer on the foundation is a related activity to the activity of connecting the transformer

to the station bus work.

The remaining terms can best be described with the aid of a diagram. On page A5 of the appendix a simplified schedule has been diagrammed (both names and time estimates are arbritrary and are intended only to illustrate terminology and the manner in which CADS schedules). The diagram shows the relationship of activities to each other with regard to the sequence in which the activities must be performed. A real time scale is included at the right of the page. On the side of the page labeled pass 1, the activities TRANSFORMER FINAL ASSEMBLY and TRANSFORMER WIRING CONNECTION are both shown as requiring the related activity TRANSFORMER PLACEMENT; however, since the activity TRANSFORMER FINAL ASSEMBLY requires more time to perform than the activity TRANSFORMER WIRING CONNECTION, the related activity TRANSFORMER PLACEMENT must be completed earlier than what is required for the activity TRANSFORMER WIRING CONNECTION. The time represented by the dash line on the diagram is known as slack time. The interval between having all the related activities complete and the latest possible starting date for an activity is defined as the slack time of the activity. For the activity TRANSFORMER WIRING CONNECTION, the slack time referring to the time scale is one day. A path is comprised of two or more activities worked in series. The activities TRANSFORMER TEST, TRANSFORMER FINAL ASSEMBLY, TRANSFORMER PLACEMENT and TRANSFORMER DELIVERY

///

form a path. Another path is defined by the activities TRANSFORMER TEST, TRANSFORMER WIRING CONNECTION, TRANSFORMER PLACEMENT and TRANSFORMER DELIVERY. It is noticed that some of the same activities occurred in the second path as were in the first path. In the first path there exists no slack time. The activity TRANSFORMER WIRING CONNECTION has slack time in the second path. The critical path is the longest path in time and contains no slack time. For the pass 1 diagram, the critical path consists of TRANSFORMER TEST, TRANSFORMER FINAL ASSEMBLY, TRANSFORMER PLACEMENT, TRANSFORMER FOUNDATION CONCRETE CURE, TRANSFORMER FOUNDATION FORM STRIP, TRANSFORMER FOUNDATION CONCRETE POUR, and TRANSFORMER FOUNDATION FORM INSTALLATION. Although the critical path is the longest path in the project, it dictates the shortest time in which the project may be completed. Only when activities in the critical path have their time shortened or are worked in parallel rather than in series can the project time be reduced.

Since in most cases the sequence in which activities are performed cannot be changed (a concrete foundation must cure before placing equipment upon it) the common way to shorten a project is to shorten the allowable time to perform each activity in the critical path. This can be accomplished by placing more manpower on the activity or by working more than one shift per day. Activities are therefore given two time estimates, normal and critical. The normal estimate is given as the time and number of men normally allowed to accomplish a given activity.

The critical estimate is based upon the time and number of men required to perform an activity when a project must be completed in minimum time. For most activities the critical time estimate will be smaller than the normal estimate, and the critical number of men estimate larger than the normal estimate. For some activities the normal and critical estimates will be the same (i.e. concrete cure).

If the left side of page A5 of the appendix is again referred to, a number pair can be observed directly below the name of each activity. The number on the left side of the slash (/) represents the normal days estimate, and the number on the right the critical days estimate. If the activities represented by the diagram were scheduled by CADS using normal estimates, the pass 1 schedule output would appear as shown on the left side of the page. Should the time required to complete this work be greater than acceptable, CADS would attempt to reschedule as shown for pass 2. The pass 2 schedule was obtained by replacing the normal estimate with the critical estimate for each activity appearing in the critical path of pass 1, (transformer test, final assembly, placement, foundation concrete cure, form strip and form installation), and rescheduling. Again if the pass 2 schedule requires too many days to complete, the activities appearing in the critical path of pass 2, (rack area backfill, grounding mat installation, and rack area excavation) would have their normal estimates replaced with critical

estimates by the CADS program. With these changes a new schedule is output as shown under pass 3, page A6. This recycling process continues until the schedule is shortened to an acceptable time or until all the activities appearing in the critical path obtained in the new pass were previously reduced to their critical estimates in an earlier pass or until the number of passes equals the limit for the maximum allowable number of passes (which can be set by the individual scheduling with the CADS program).

In order to describe some of the output features of CADS, the following definitions and explanations are made.

Activity Index Number. This is a computer generated activity identification number, which is unique for each particular activity. This activity number is assigned when all activities are sorted, alphamerically by summary, during processing. The activity index number will normally change for subsequent schedule recasting. It is usually used in locating a specific activity when the schedule outputs are being examined.

File Number. This is an arbitrarily chosen activity identification number assigned by the scheduler, and is used for identifying an activity external to the computer. This number does not change for subsequent schedule recasting and is therefore useful for activity data modifications.

The output features of CADS are:

Summary - Activity Table. This table is printed as the first item in the schedule and lists all the summary numbers and summary names used, with the number of activities assigned to each summary. The heading contains the date at which the schedule is made, the project name, and the page number. Next, listed in numerically ascending order are the summary numbers with their respective summary names and assigned activities (see appendix, page A11).

Critical - Path Table. This table provides a list of all activities in the critical path for the specified pass number. The heading shows the date at which the schedule is made, the name of the project, the pass and the page number. Next, a total indicates the number of activities comprising the critical path, as well as the number of working days for the critical path. If the computer generated schedule does not terminate within the required completion time period, a message is printed out showing the pass number and the number of working days by which the schedule "Overshoots" the project completion date. In addition to the activity name, the index number, file number and time estimate are also listed. The critical path activities are arranged so that activities occurring towards the beginning of the project are listed first. (See appendix, pages A12-A13).

If rescheduling occurs, then subsequent critical-path tables will be printed for each pass.

Alphabetic Schedule. The alphabetic schedule alphamerically lists all activities, with their related activities, for each summary. The heading includes the date at which the schedule is made, the name of the project, the summary name, the pass number, and the page number. Information included for each activity in this schedule is the activity index number, the activity name, the file number, the normal time and manpower estimates, the critical time and manpower estimates, the computed, if not initially estimated, total man-hour requirements, the slack time, the start date, and the finish date. Information pertaining to related activities is indented. Time and manpower estimates actually used in the schedule are marked with an asterisk. Any activity previously completed or finished will be indicated by the word "Completed" in the start and finish columns. (See appendix, pages A14-A16).

Chronological (Bar Chart) Schedule. The graphical type of schedule lists activities chronologically by starting date for each summary. The heading includes the date at which the schedule is made, the name of the project, the summary name, and the page number. The maximum period for which scheduling information can be provided on one page is four months. In addition to the activity name, the activity manpower requirement and slack time are also shown. Near the top of the page, the months and years under consideration are indicated. The next two lines show actual calendar work days. Weekends and holidays are

omitted. The first line is the first digit, or tens position of the calendar day, and the second line is the second digit or units position of the calendar day. The "X's" span the time interval during which the activity is to be performed. A "C" occurring on the last calendar day of a page indicates that the "X" sequence is continued into a subsequent time period. Therefore, the same activity will occur on another page of the schedule. An activity which has been completed will not appear in the chronological schedule.

At the end of the list of a summary containing all activities for a specific time interval, usually four months, a "total manpower requirements" chart is printed. The values indicate the total number of men required by the group represented in the summary, for any particular date during the interval. The first, second, and third lines indicate the hundreds, tens, and units positions, respectively, of the manpower value. (See appendix, pages A17-A18).

HOW DOES THE SCHEDULER AND/OR ENGINEER DESCRIBE THE PROJECT TO THE CADS SYSTEM?

The schematic approach (arrow diagramming) for collecting schedule data for the CADS program was avoided because of the difficulties in showing the sequence relationships between activities, and because of the limited number of activities which can be shown on a single sheet of paper.

The collection of data for a project schedule is

begun by examining the finished product and asking the questions: Which components make up the finished product, and what is the last activity performed on each of these components before they are declared complete?

This initial list of activities is called the end-of-project activity list. It is not necessary to include in this list any activity which is a prerequisite to any other activity.

Each end-of-project activity is then examined to determine its immediate prerequisite activities. Then the immediate prerequisite activities to the activities preceding the end-of-project activities are sought, etc. The activities which must immediately precede an activity are called that activity's related activities.

The order of performance of activities (starting with a particular end-of-project activity) for most installations is as shown below. Indented activities are related (immediate prerequisites) to the non-indented activity directly above.

COMPONENT TEST
  COMPONENT INSTALLATION

COMPONENT INSTALLATION
  COMPONENT MATERIAL DELIVERY
  COMPONENT INSTALLATION DRAWING REVIEW BY CONSTRUCTION
  COMPONENT AREA PREPARATION

COMPONENT MATERIAL DELIVERY
  COMPONENT BID AWARD TO DELIVERY TIME

COMPONENT BID AWARD TO DELIVERY TIME
COMPONENT AWARD OF CONTRACT, APPROVAL

COMPONENT AWARD OF CONTRACT, APPROVAL
COMPONENT BID EVALUATIONS

COMPONENT BID EVALUATIONS
COMPONENT BID ADVERTISEMENT TIME

COMPONENT BID ADVERTISEMENT TIME
COMPONENT SPECIFICATION APPROVAL

COMPONENT SPECIFICATION APPROVAL
COMPONENT SPECIFICATION PREPARATION

COMPONENT SPECIFICATION PREPARATION
(The prerequisite of preparing any specification is generally the engineering of some key drawing (i.e., one line wiring, design equipment, or structural steel) and/or the preparation of an authorization.)

COMPONENT INSTALLATION DRAWING REVIEW BY CONSTRUCTION
COMPONENT INSTALLATION DRAWING APPROVAL

COMPONENT INSTALLATION DRAWING APPROVAL
COMPONENT INSTALLATION DRAWING REVIEW BY DESIGN ENGINEERING

COMPONENT INSTALLATION DRAWING REVIEW BY DESIGN ENGINEERING
COMPONENT INSTALLATION DRAWING DRAFTING CHECK

COMPONENT INSTALLATION DRAWING DRAFTING CHECK
COMPONENT INSTALLATION DRAFTING

COMPONENT INSTALLATION DRAFTING
COMPONENT INSTALLATION DESIGN ENGINEERING

COMPONENT INSTALLATION DESIGN ENGINEERING
(The prerequisite of engineering varies. It can be one or all of the following: authorization prepared, other engineering complete, other drawings prepared, and manufacturer's drawing prepared.)

COMPONENT AREA PREPARATION
(e.g. installation of equipment board, erection of a building, concrete cure, etc.)

The collection of data as shown in the above section for a project schedule is accomplished by describing in order starting with the end-of-project activities, step by step, the activities necessary for the construction of a project. Once the construction phase of the data collection has been completed far enough to require the delivery of material or drawings, the steps for material procurement and obtaining drawings are almost always the same for each item in most organizations.

After the data breakdown has been decided, time estimates are given to each activity. The time estimate is based on how long it will take to complete a given activity if all of its related (or prerequisite) activities are completed first. The time estimate can be given in working days with the manpower (number of men) estimate optional or as a total man-hour estimate with the manpower estimate required. If an activity has previously been completed, that information can also be entered.

The above data collection method is based on the following hypotheses: A group or person charged with the responsibility for performing a given activity (particularly if the activity is similar to one performed previously) can reasonably answer at least two questions about the activity. First, what is needed (related or prerequisite activity list) before the activity can be started? Second, if everything that is needed to start the activity is available, how long

will it take to perform the activity?

Once each activity making up the project has been defined (given a time estimate and a related activity list) and keypunched on IBM cards, the scheduler only needs to specify the starting date (normally today) and the target date for the project. This information, together with the project title, summary names (all on IBM cards) and the CADS program are all that is necessary for running the program on the IBM 1620 computer and obtaining a schedule.

Some of the characteristics of the CADS program are as follows:

If an activity is not listed as a related activity to any other activity, it will become an end-of-project activity and have a completion date equal to that of the completion date of the entire project.

If an activity is listed as a related activity of several other activities, its completion date will precede the starting date of the earliest starting activity requiring it as a related activity.

Every related activity must appear as a defined activity.

CADS places the begin date of all activities as close to the end-of-project date as possible within the limits set by the activity time estimates and their prerequisites.

In the process of collecting data for a project schedule, one might wonder how fine of a data breakdown is necessary. The

answer to this question is that the detailed breakdown of data should be made fine enough to allow time estimates to be made without having to estimate dead time for each activity. Dead time as referred to here means the time wasted on an activity, stopped because of lack of information and/or material necessary to complete the activity. If the input data is collected in sufficient detail, each activity's related (prerequisite) activities are scheduled to be completed before the activity requiring them is scheduled to be started. Thus, assuming reasonably accurate data and an adherence to schedule dates, the situation where an activity could have been completed earlier had it only been known that it was needed by a certain date can be prevented.

Again, as a defense against the argument that too much data just confuses the schedule and makes it impossible to follow, it must be pointed out that the individual groups performing the activities are given individual summary schedules containing only the activities assigned to their particular group and not the entire schedule output. Furthermore, once the data collected for a project schedule has been thoroughly checked out and the target date established, it is not necessary for management to try to follow the entire schedule output.

Management, in order to keep watch over a project, needs to know whether or not the schedule is being adhered to. Schedulers and coordinators also need this information in order

to know if a schedule needs to be updated. This can be accomplished by having the schedulers collect periodically (i.e., monthly) from the individual groups the progress of all activities completed, active or started during the month. The number of activities completed, active or started during the month is very small. The number of these activities which are not progressing according to the schedule is even smaller. Therefore, with an effective reporting system, management needs only to be concerned with a few activities during any month.

To further simplify the problem when one or more activities deviate appreciably from the schedule, the updated data can be automatically rescheduled by the CADS program in a matter of minutes. The CADS program adjusts the schedule through its ability to find the critical path, modify the critical path activity time estimates, and recast a schedule iteratively until the project is brought back to the original target date. The steps taken by the program to adjust the schedule are printed out in concise form. CADS updates prints, proofs and readies the complete, detailed schedule for distribution automatically.

WHAT ARE THE SUPPORTING SYSTEMS AND ERROR CHECKING FEATURES OF THE SYSTEM?

Since the Department has an IBM 1620 computer, several programs have been written for the machine which provide a convenient method for CADS data preparation and checking. One

program of this type can generate prerequisite activities in a sequence for input to the CADS program when given certain specific activities whose sequence or trace is a known or fixed procedure in the Department. Three commonly used sequences are drawing, specification, and bill of material preparation. Given an equipment delivery because of formal bid or bill of material requisition, or a drawing delivery, this program will generate activities in the proper sequence down to and including engineering. Other programs can delete, change, or list specific cards in a CADS data file. In order to provide a rapid method for producing data for similar projects programs have been written which will reproduce and renumber CADS input data files.

A 1620 program is also available which will generate block diagrams of the project showing the activity relationships. This program uses the CADS input data to generate these diagrams.

Some error checking features have been developed in the CADS computer program in order to facilitate data debugging. One feature provides a check on activities which have either not been defined or have been improperly defined. When this occurs, a message "Found no J num for----------", is printed out. Another common error is feedback. Feedback is an improperly defined relationship between two or more activities. When this occurs two or more activities are prerequisites of

each other, either directly or indirectly. Feedback is noted when a list of affected activities is printed out with no subsequent schedule information. A sub-program has been written which will handle the feedback-affected activities and produce a list of activities showing the paths which are directly involved in the feedback. As can be seen, these features provide means by which data handling and correction can be reduced to a minimum.

EDUCATION REQUIREMENTS FOR IMPLEMENTING CADS

The problem in securing and training personnel to implement an automated scheduling program deserves comment. As was stated above, the original programming and schedule preparation was carried out by special studies engineers familiar with computers, but the staff scheduling engineers who now run the system had little or no knowledge of computers. The engineering scheduling personnel have been sent to short computer orientation and training sessions to acquire basic operating knowledge. In addition, good success has been obtained from brief training periods of clerical help in a step-by-step process over a period of months. In the operation of any computing system, the main key to success is finding interested, careful personnel.

Since the special studies group provides consulting service when difficult or unusual circumstances are encountered, little knowledge is required by the schedulers concerning the operating characteristics of the IBM 1620 which is used for

the major portion of the Department's automated scheduling program.

EQUIPMENT

A typical 1500 activity CADS schedule requires about two hours of IBM 1620 time. In addition to the IBM 1620, another computer, the IBM 1401, is used for report writing. These two machines are available within the Department. Also, the Department has a well-equipped card tabulating section, which provides services for reproducing, sorting, collating, and other preliminary card handling procedures for CADS.

FUTURE

Planned modifications and additions for the CADS system are as follows:

1. The inclusion of job or activity cost data with the aim of developing a program which would schedule a project for a minimum overall cost.

2. The establishment of an automated division-wide manpower allocation system for scheduling of major projects simultaneously.

3. The inclusion of features for special job progress reports, to be provided for management use.

CONCLUSIONS

The Department has applied an automated scheduling system to the task of scheduling and coordinating the design and construction of several large substations and to the task

of scheduling and coordinating the construction and preliminary operations of several large steam generating units currently under construction. This scheduling system has been operating successfully since the early part of 1961 and is now an integral part of the Design and Construction Division's operations.

Critical Path and PERT
BIBLIOGRAPHY
1961-1962

1. Berman, H. Try critical path method to cut turnaround time 20 percent. Hydrocarbon Processing and Petroleum Refiner, 41:135-8, January 1962.

2. Builders bone up on critical path at MIT. *Engineering News-Record, 168:19-20, June 28, 1962.

3. CPM enthusiasts claim bonus benefits. *Engineering-News-Record, 169:42-44, August 9, 1962.

4. Cabell, C. P. Integrated programming; simplified version of PERT. Plant Engineering 16:106-12, September 1962.

5. Christensen, B. M. Network models for project scheduling. Machine Design, 34:114-18, May 10, 1962; 34:173-7, May 24, 1962; 34:132-8, June 7, 1962; 34:155-60, June 21, 1962; 34:105-11, July 5, 1962; 34:136-40, July 19, 1962.
   May 10: Planning phase--initial steps in setting up network models with special emphasis on critical-path method. May 24: Preliminary schedule phase. June 7: Advanced scheduling phase; determining critical points; choosing schedule. June 21: First steps in working our samples project. July 5: Computer routines for planning and scheduling problems of any practical size. July 19: Establishing specific plan of implementation.

6. Cosinuke, W. Critical-path technique for planning and scheduling. *Chemical Engineering, 69:113-18, June 25, 1962.

7. Critical path method; new tool for job management. *Engineering News-Record, 166:25-7, January 26, 1961.

8. Driessnack, H. H. PERT on C-141. Aerospace Management, 5:32-5, August 1962.

9. Eisner, H. Generalized network approach to the planning and scheduling of a research project; PERT technique. Operations Research, 10:115-25, January 1962.

10. Glaser, L. B. Critical path planning and scheduling; application to engineering and construction. Chemical Engineering Progress, 57:60-5, November 1961.

*Magazine titles marked with an asterisk may be found in the Department library

11. Hawthorne, R. Pert and pep; useful tools or timewasters? Space/Aeronautics, 36:56-8, August 1961.

12. Healy, T. L. Activity subdivision and PERT probability statements. Operations Research, 9:341-50, May-June, 1961.

13. Huish, H. A. Keeping critical path up-to-date is no problem with a computer. *Plant Management and Engineering, 24:15-19, April 1962.

14. Industry borrows control tool that's closing missle gap. *Iron Age, 190:62-4, July 5, 1962.

15. Kast, W. G. Critical path method ideal tool for plant construction. Hydrocarbon Processing and Petroleum Refiner, 41:123-30, February 1962.

16. Kelley, J. E. Critical-path planning and scheduling: Mathematical basis. Operations Research, 9:29-320, May-June 1961.

17. Kruse, M. E. Streamline your maintenance planning with critical path scheduling. *Plant management and Engineering, 23:49-51, December 1961.

18. Loeber, N. C. PERT for small projects. Machine Design, 34:134-9, October 25, 1962.

19. Lundenheimer, E. L. Use critical path method to plan complex projects. *Power Engineering 66:37-40, September 1962.

20. Lynch, C. J. Plan projects scientifically with critical-path scheduling. *Product Engineering, 34:92-6, September 18, 1961.

21. Martin, N. M. Critical-path method expedites IBM projects. *Engineering News-Record, 169:34-6, July 5, 1962.

22. Mauchly, J. W. Critical-path scheduling. *Chemical Engineering, 69:139-54, April 16, 1962.

23. Meyers, R. E. Critical path scheduling as applied to refinery turn-arounds; abstract. Chemical Engineering Progress, 58:104, July 1962

*See Footnote on page A1.

24. Pearlman, J. Engineering program planning and control through use of PERT. Institute of Radio Engineers. Transactions on Engineering Management, EM-7:125-34, December 1960.

25. Reeves, E. Critical-path speeds refinery revamp. Canadian Chemical Processing, 44:74-6+, October 1960.

26. Schureman, L. R. 'Critical path' as job scheduling and control aid. Roads and Streets, 105:66-8, 70, June 1962.

27. Stagg, G. W. et al. PERT schedules manpower. *Electrical World, 158:36-7, July 30, 1962.

28. Stegber, C. B. Scheduling projects by critical path. *Electronics, 35:56-7, March 2, 1962.

29. Steinfield, R. C. Critical path saves time and money. *Chemical Engineering, 67:148, 150-2, November 28, 1960.

30. Whalen, J. M. Adding performance control to cost control. National Association of Accountants. Bulletin, 43:67-74, August 1962.

31. Young, L. H. Now industry schedules by computer; PERT (Project evaluation and review technique) on critical path method (CPM). Control Engineering, 9:16-18, January 1962.

32. Proceedings of the Seventh Annual Engineering and Operations Workshop, American Public Power Association, Phoenix Arizona, January 1963.

33. Proceedings of the AIPE-ASME Plant Engineering Conference, Los Angeles, California April 1964

*See Footnote on Page A1

SUMMARY INDEX

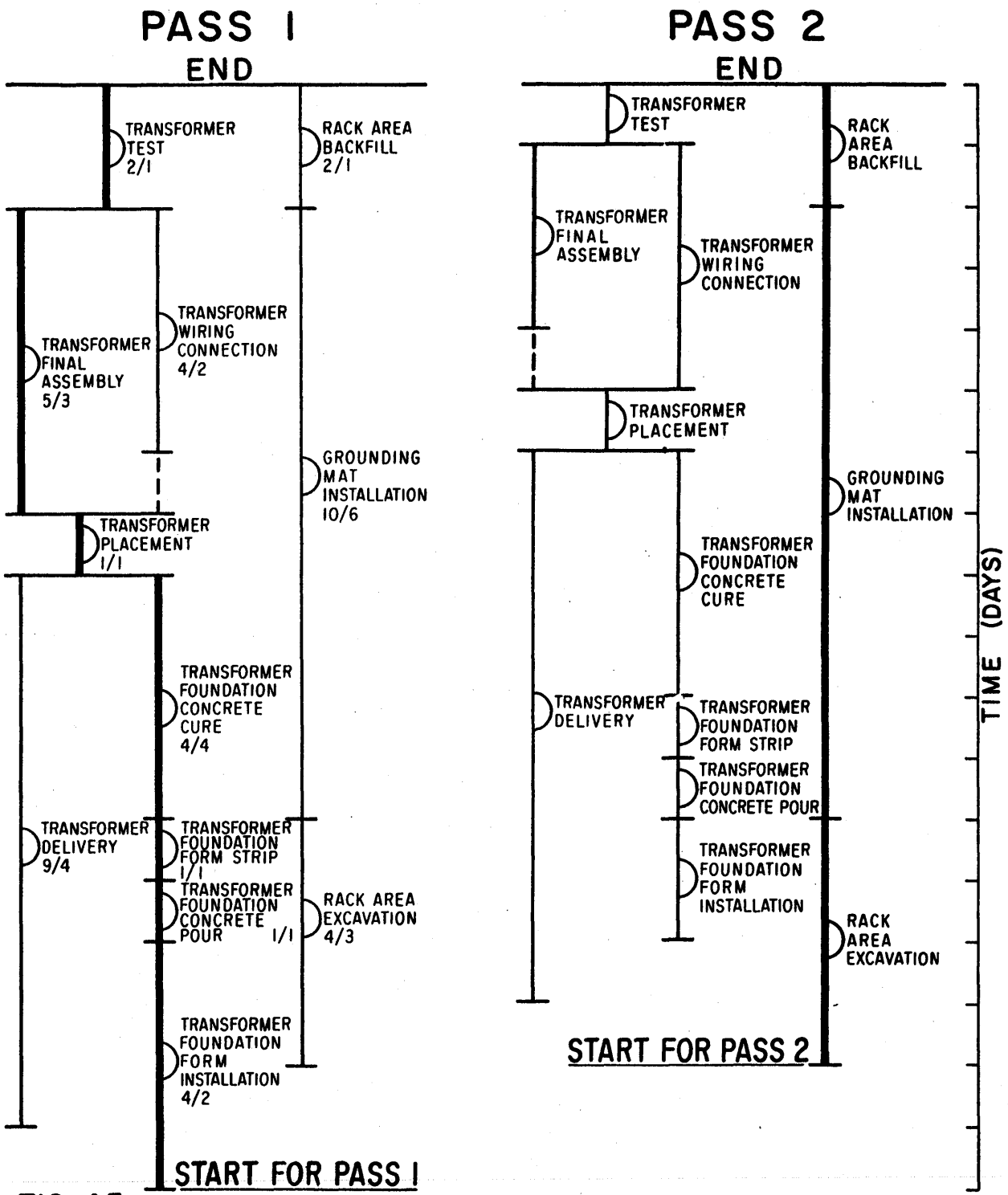


FIG. A5

MULTI-PASS SCHEDULE COMPRESSION

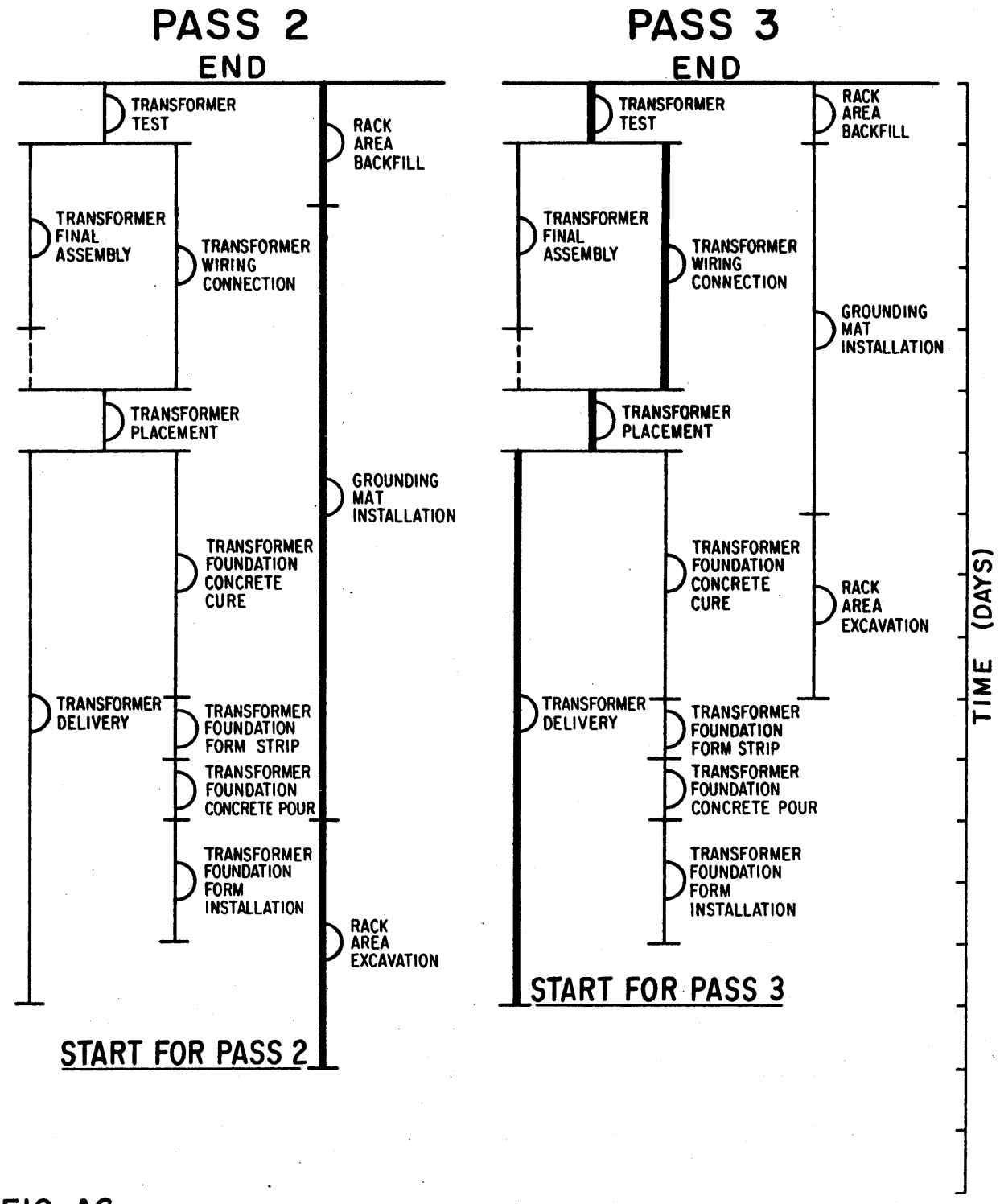


FIG. A6

MULTI-PASS SCHEDULE COMPRESSION

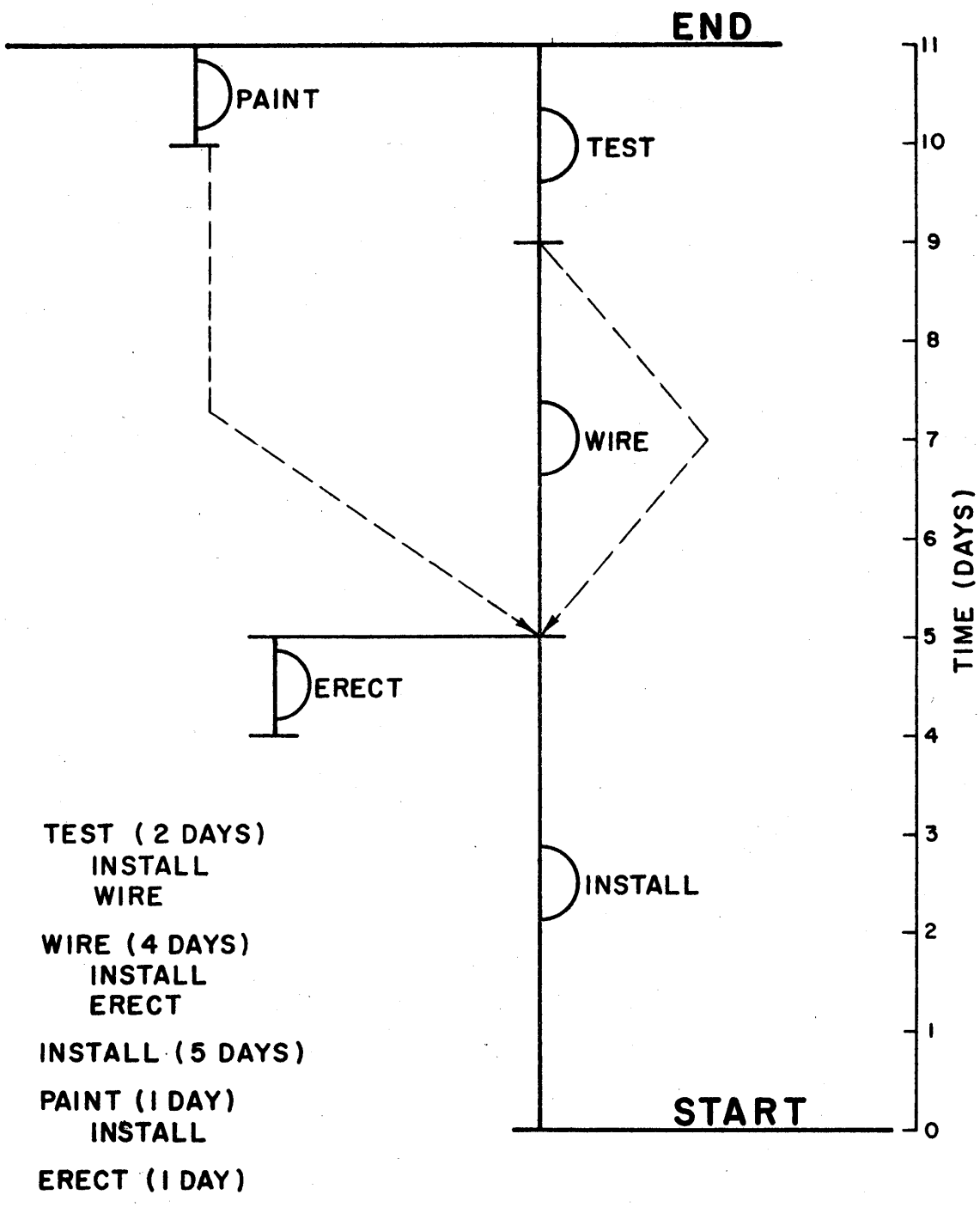


FIG. A7 GRAPHICAL REPRESENTATION OF CADS SCHEDULING TECHNIQUE (BASED ON ACTIVITY TIME AND PREREQUISITE LIST)

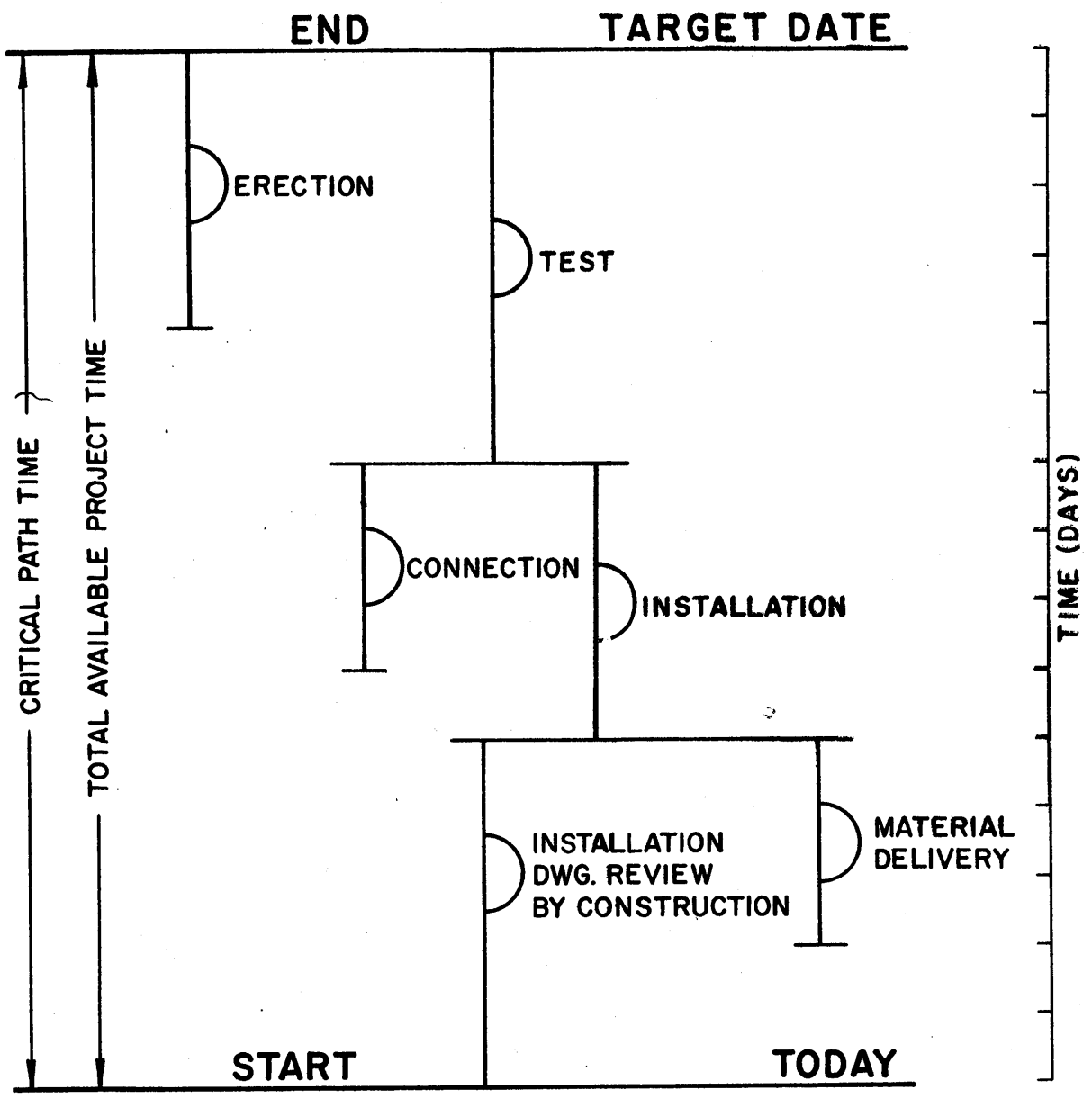


FIG. A 8 SCHEDULE WITH CRITICAL PATH TIME EQUAL TO TOTAL AVAILABLE PROJECT TIME

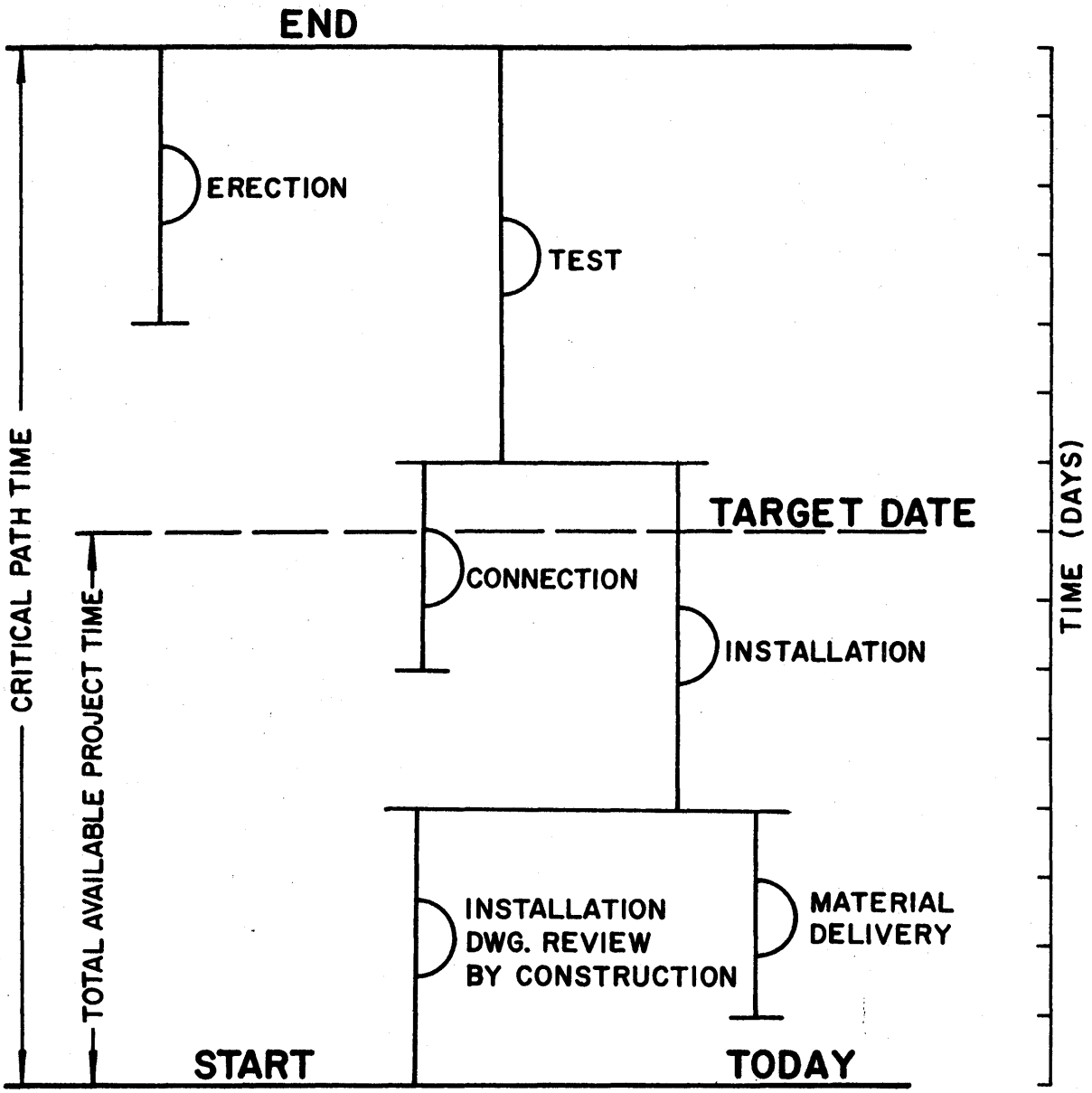


FIG. A9 SCHEDULE WITH CRITICAL PATH TIME GREATER THAN THE TOTAL AVAILABLE PROJECT TIME

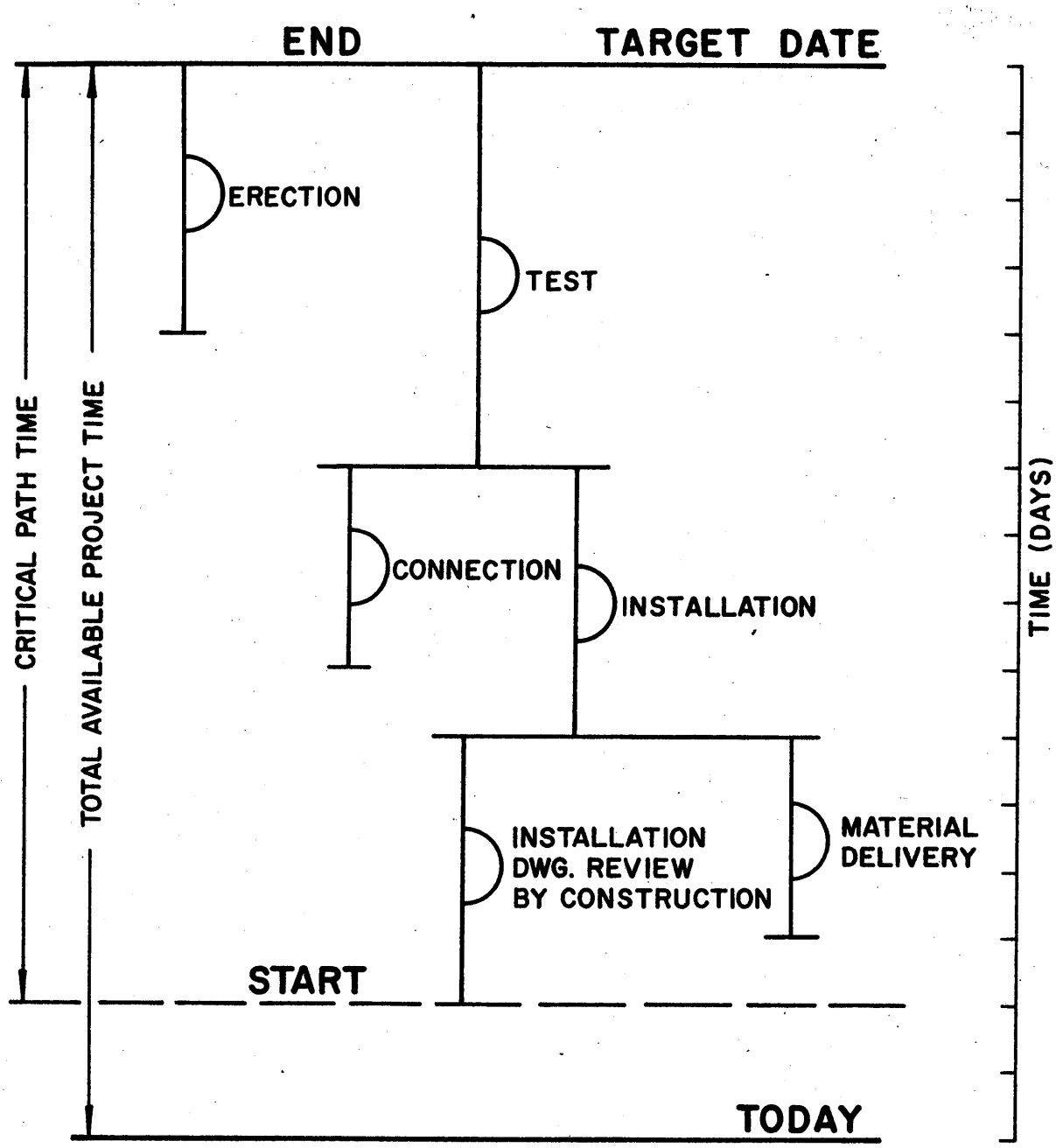


FIG. A10 SCHEDULE WITH CRITICAL PATH TIME LESS THAN THE TOTAL AVAILABLE PROJECT TIME

SUMMARY NUMBER	SUMMARY NAME	NUMBER OF ACTIVITIES
20	PROJECT MATERIAL DELIVERY	155
21	CONTRACT WORK	24
24	CONCRETE PLACEMENT	18
25	PLUMBER	3
26	EXCAVATION & GRADING	11
28	ELECTRICAL INSTALLATION	116
32	NONSENSE	21
33	CONSTRUCTION FACILITIES	8
37	COMMUNICATIONS	4
45	MECHANICAL INSTALLATION	33
46	PIPING INSTALLATION	7
49	MECHANICAL INSTRUMENT INSTL	9
57	STEEL FABRICATION & ERECTION	5
58	START-UP OPERATIONS	46
86	NUCLEAR CONTRACT WORK	170
87	NUCL&RADIOLOGICAL INSTR INSTL	13
88	NUCLEAR START-UP OPERATIONS	28
89	NUCLEAR PROJECT MATERIAL DLVRY	119
90	NUCLEAR FUEL DELIVERY	2

138

All

CRITICAL PATH

35 ACTIVITIES COMPRISE THE CRITICAL PATH TOTALING 1395 WORKING DAYS

PASS NO. 1 SCHEDULE OVERSHOOTS DESIRED PROJECT COMPLETION DATE BY 14 WORKING DAYS

INDEX NO.	ACTIVITY NAME	FILE NO.	DAYS
463	AEC PUBLIC HEARING RECONVENES	570044	15
346	TIME SPACER,CP EFF TO PUB HRNG	570045	104
466	CONSTRUCTION PERMIT EFFECTIVE	570046	1
604	SIGN WEST CONTRACT FOR S&W ENG	570039	1
464	AUTHORIZE TITLE 1 ENGINEERING	570040	66
585	S&W CONTAIN STRUCTURE DESIGN	570041	55
470	DBL LINER#BID TO PURCHASE ORDR	570042	66
469	DBL LINER PLATE FABRICATION	570043	132
576	RX CONTA STRUC FOUN MAT IN	570001	44
540	REACCONTASTRU BOT DBL LINER IN	570003	55
665	REACCONTA BOT LINER FREON TEST	570004	6
543	REACONTAVRTDBLLNR&POPCRNCONCIN	570011	66
552	REACT CONTA INTERIORCONC PHBIN	570013	52
569	REACTOR CRANE IN	660017	11
539	REACCONTADMEDBLLNR&POPCRNCONIN	570014	198
667	REACT CONTA FREON TEST	570015	11
545	REACT AUX SYS EQ & PPG PHA IN	520002	154
666	REACT CONTA AIR TEST	570032	11
519	RC PPG HANGERS LOOPA PHB IN	580018	33
531	RCPPGFIT&WELD&PRECLEANLOOPAPHB	580015	33
524	RCP MTR LOOP A IN	500007	6
514	RC PPG COLD SPRING&FINAL WELD	580004	55
664	RC SYSTEM FILL & COLD FLUSH	500053	6
663	RC SYS INITIAL HYDRO TEST	500054	6
623	THERM INSULA FIELD JOINTS IN ◻	500003	6
658	NUCL SYSTEMS HOT FUNCT TESTING	650003	22

139

A12

INDEX NO.	ACTIVITY NAME	FILE NO.	DAYS
668	REMOVE RV HD CLEAN & FLUSH SYS	660006	6
468	CORE LOADING & ASSEMBLY	660007	15
575	RV TOP INTERNALS IN	660008	11
669	RV HD IN 2 & FINAL HYDRO TEST	660009	11
650	CONTROL ROD TESTING	660010	6
654	INITIAL CRITICAL& 0 PWRTESTING	650002	19
442	STEAM LINES STEAM BLOW	200054	5
329	TIME SPACER	200102	14
662	PWR GENERATION TESTING 10-100%	650001	93

ELECTRICAL INSTALLATION

¤PREREQUISITE ACTIVITIES ARE INDENTED¤ ¤*INDICATES ESTIMATE USED¤

INDEX NO.	ACTIVITY NAME	FILE NO.	NORMAL DAYS	NORMAL MEN	CRITICAL DAYS	CRITICAL MEN	TOTAL MANHRS	DAYS SLACK	START DATE	FINISH DATE
252	ISOLATED PH 22KV BUS SUPPORTIN	040030	10*	0*	10	0	0	0	22 MAY 1970	4 JUN 1970
68	ISOLATED PH 22KV BUS SUPPORTDL	040031								
253	ISOLATED PHASE 22KV BUS IN	040028	66*	0*	66	0	0	0	5 JUN 1970	4 SEP 1970
69	ISOLATED PHASE 22KV BUS DL	040029								
252	ISOLATED PH 22KV BUS SUPPORTIN	040030								
254	LUBE OIL PMP MTR CND&WIR IN	040238	10*	4*	9	4	320	0	29 JUL 1970	11 AUG 1970
160	LUBE OIL PMP&RESERVOIR INSTL	180007								
255	MTR HTR DISTR EQUIP IN&WIR	040137	22*	0*	22	0	0	0	29 JUL 1970	27 AUG 1970
185	NUCSERVICE&CONTROL HOUSE CONST	010199								
76	MTR HTR DISTR PNL DL	040138								
77	MTR HTR RELAY CABINET DL	040139								
78	MTR HTR RELAYS DL	040140								
79	MTR HTR SUPPLY TRANSF DL	040141								
256	NSC BLDG CABLE RISER IN	040056	10*	0*	10	0	0	0	14 AUG 1969	27 AUG 1969
185	NUCSERVICE&CONTROL HOUSE CONST	010199								
80	N&C BLDG CABLE RISER MATL DL	040057								
257	NSC BLDG CABLE TRAY SYS IN	040050	33*	0*	33	0	0	0	14 JUL 1969	27 AUG 1969
185	NUCSERVICE&CONTROL HOUSE CONST	010199								
81	N&C BLDG CABLE TRAY DL	040051								
258	NSC BLDG GRD GRID&ELECTRODE IN	040044	5*	0*	5	0	0	0	23 SEP 1968	27 SEP 1968
184	N&C BLDG CABLE VAULT IN	010200								
82	NSC BLDG GRD GRID&ELECTRODE DL	040045								
259	NSC BLDG LIGHTING CONDUIT IN	040046	10*	0*	10	0	0	0	21 JUL 1969	1 AUG 1969
185	NUCSERVICE&CONTROL HOUSE CONST	010199								
83	NSC BLDG LIGHTING CONDUIT DL	040047								
260	NSC BLDG LIGHTING IN&WIR	040048	22*	0*	22	0	0	0	4 AUG 1969	3 SEP 1969
259	NSC BLDG LIGHTING CONDUIT IN	040046								
84	NSC BLDG LIGHTING EQUIP DL	040049								
261	NSC BLDG TERM RM CABLE TRAY IN	040054	22*	0*	22	0	0	0	4 AUG 1969	3 SEP 1969
185	NUCSERVICE&CONTROL HOUSE CONST	010199								
85	NSC BLDG TERM RM CABLE TRAY DL	040055								
262	NSC BLDG TEST CIRC PL&EQIN&WIR	040060	33*	0*	33	0	0	0	14 JUL 1970	27 AUG 1970
185	NUCSERVICE&CONTROL HOUSE CONST	010199								
86	NSC BLDG TEST CIRC PL&EQ DL	040061								
263	NSC BLDG VAULT CABLE TRAY SYIN	040052	22*	0*	22	0	0	0	27 JAN 1970	26 FEB 1970
185	NUCSERVICE&CONTROL HOUSE CONST	010199								
184	NSC BLDG CABLE VAULT IN	010200								
87	NSC BLDG VAULT CABLE TRAY DL	040053								
264	NSC CABLE TERMINAL EQ IN&WIR	040064	22*	0*	22	0	0	0	27 JAN 1970	26 FEB 1970
185	NUCSERVICE&CONTROL HOUSE CONST	010199								

A14

141

ELECTRICAL INSTALLATION

¤PREREQUISITE ACTIVITIES ARE INDENTED¤ ¤*INDICATES ESTIMATE USED¤

INDEX NO.	ACTIVITY NAME	FILE NO.	NORMAL DAYS	NORMAL MEN	CRITICAL DAYS	CRITICAL MEN	TOTAL MANHRS	DAYS SLACK	START DATE	FINISH DATE
88	NSC CABLE TERMINAL EQ DL	040065								
265	OFFICE&GUARD BLDG LIGHT CON IN	040208	15*	0*	15	0	0	0	20 FEB 1969	12 MAR 1969
91	OFFICE&GUARD BLDG LIGHT CON DL	040209								
164	OFF BLDG&GUARD HOUSE CONST	020030								
266	OFFICE&GUARD BLDG LIGHT IN&WIR	040210	44*	0*	44	0	0	0	13 MAR 1969	13 MAY 1969
265	OFFICE&GUARD BLDG LIGHT CON IN	040208								
92	OFFICE&GUARD BLDG LIGHT EQ DL	040211								
267	PA&WDB UTILITY SERVICE FA IN	580022	15*	0*	15	0	0	952	23 APR 1969	13 MAY 1969
268	PRESSURIZER HTR FA CTL&PWR WIR	040199	10*	0*	10	0	0	0	27 FEB 1970	12 MAR 1970
263	NSC BLDG VAULT CABLE TRAY SYIN	040052								
261	NSC BLDG TERM RM CABLE TRAY IN	040054								
256	NSC BLDG CABLE RISER IN	040056								
217	CABLE TUNNEL CABLE TRAY IN	040058								
264	NSC CABLE TERMINAL EQ IN&WIR	040064								
509	PRESSURIZER IN	500058								
269	PRIMARYCONTA ELEC SERV FA IN	570034	1*	0*	1	0	0	1174	12 MAR 1970	12 MAR 1970
270	PWR CONTROL RELIEF VALVE IN&WI	040172	10*	0*	10	0	0	0	14 AUG 1970	27 AUG 1970
93	PWR CONTROL RELIEF VALVE EQ DL	040173								
271	REACT COOL PP1A FA CTL&PWR WIR	040181	10*	0*	10	0	0	0	27 FEB 1970	12 MAR 1970
263	NSC BLDG VAULT CABLE TRAY SYIN	040052								
261	NSC BLDG TERM RM CABLE TRAY IN	040054								
256	NSC BLDG CABLE RISER IN	040056								
217	CABLE TUNNEL CABLE TRAY IN	040058								
264	NSC CABLE TERMINAL EQ IN&WIR	040064								
524	RCP MTR LOOP A IN	500007								
272	REACT COOL PP1B FA CTL&PWR WIR	040182	10*	0*	10	0	0	187	27 FEB 1970	12 MAR 1970
525	RCP MTR LOOP B IN	500045								
273	REACT COOL PP1C FA CTL&PWR WIR	040183	10*	0*	10	0	0	155	27 FEB 1970	12 MAR 1970
526	RCP MTR LOOP C IN	500034								
274	REACT COOL PP1D FA CTL&PWR WIR	040184	10*	0*	10	0	0	77	27 FEB 1970	12 MAR 1970
527	RCP MTR LOOP D IN	500024								
275	SAFETYINJ PP MTR1A CTL&PWR WIR	040189	10*	0*	10	0	0	36	4 MAY 1970	15 MAY 1970
586	SAFETY INJECTION PMP 1A INSTL	530001								
276	SAFETYINJ PP MTR1B CTL&PWR WIR	040190	10*	0*	10	0	0	36	4 MAY 1970	15 MAY 1970
587	SAFETY INJECTION PMP 1B INSTL	530002								
277	SAFETYINJ PP MTR1C CTL&PWR WIR	040191	10*	0*	10	0	0	36	4 MAY 1970	15 MAY 1970
588	SAFETY INJECTION PMP 1C INSTL	530003								
278	SHOP&WREHSE B LIGHT CONDUIT IN	040204	22*	0*	22	0	0	0	17 APR 1969	16 MAY 1969
411	SHOP&WREHSE SUPERSTRUCTURE IN	020006								

A15

1472

ELECTRICAL INSTALLATION

¤PREREQUISITE ACTIVITIES ARE INDENTED¤ ¤*INDICATES ESTIMATE USED¤

INDEX NO.	ACTIVITY NAME	FILE NO.	NORMAL DAYS	NORMAL MEN	CRITICAL DAYS	CRITICAL MEN	TOTAL MANHRS	DAYS SLACK	START DATE	FINISH DATE
100	SHOP&WHEHSE B LIGHT CONDUIT DL	040205								
279	SHOP&WREHSE BLDG LIGHT IN&WIR	040206	39*	0*	33	0	0	0	19 MAY 1969	3 JUL 1969
278	SHOP&WREHSE B LIGHT CONDUIT IN	040204								
101	SHOP&WHEHSE BLDG LIGHT EQ DL	040207								
280	STATOR COOLING SYSTEM IN&WIR	040015	33*	0*	33	0	0	0	22 JUL 1970	4 SEP 1970
105	STATOR COOLING SYS EQUIP DL	040016								
281	STEAM TEMP INSTR&CONTROL IN&WI	040170	44*	0*	44	0	0	194	11 JUN 1970	11 AUG 1970
108	STEAM TEMP INSTR&CONTROL EQ DL	040171								
282	TURB INSTR WIR-TURB TERMCAB IN	040239	30*	4*	27	4	960	0	17 JUL 1970	27 AUG 1970
111	TURB INSTR WIR-TURB TERMCAB DL	040240								
178	TURB TERMINAL CAB INSTL	180015								
283	TURB INSTRUMENT&CONTROL IN&WIR	040004	44*	0*	44	0	0	0	29 JUN 1970	27 AUG 1970
112	TURB INSTRUMENT&CONTROL EQ DL	040005								
284	TURB-GEN CABLE TRAY SYS IN	040040	66*	0*	66	0	0	0	26 MAY 1969	27 AUG 1969
192	TURBPLT ELEV 35 CONC DECK IN	010113								
118	TURB-GEN CABLE TRAY DL	040041								
285	TURBGEN GRD GRID&ELECTRODE IN	040042	33*	0*	33	0	0	0	25 JUL 1968	10 SEP 1968
119	TURBGEN GRD GRID&ELECTRODE DL	040043								
286	TURBINE REGULATING CAB IN&WIR	040021	22*	0*	22	0	0	31	20 JUL 1970	18 AUG 1970
62	GENERATOR MAJOR PARTS DL	040001								
287	TURBPLT BLDG LIGHT CONDUIT IN	040036	75*	0*	75	0	0	0	22 JAN 1970	7 MAY 1970
121	TURBPLT BLDG LIGHT CONDUIT DL	040037								
414	TURBPLT SUPERSTR WALLS&ROOF IN	180039								
288	TURBPLT BLDG LIGHTING IN&WIR	040038	180*	0*	180	0	0	0	8 MAY 1970	22 JAN 1971
287	TURBPLT BLDG LIGHT CONDUIT IN	040036								
122	TURBPLT BLDG LIGHT EQUIP DL	040039								
289	TURBPLT GRDGRID IN	040222	15*	0*	15	0	0	0	15 FEB 1968	7 MAR 1968
355	TURBPLT EX UNWATERING SYS IN	000006								
125	TURBPLT GRDGRID DL	040225								
290	YARD CONDUIT&MANHOLE SYSTEM IN	040214	792*	0*	792	0	0	0	31 MAY 1967	15 JUL 1970
130	YARD CONDUIT DUCT MATL DL	040215								
291	120V INSTR SUPPLY PL&EQ IN&WIR	040134	22*	0*	22	0	0	0	8 JAN 1970	6 FEB 1970
185	NUCSERVICE&CONTROL HOUSE CONST	010199								
131	120V INSTR SUPPLY PL&EQ DL	040135								
132	120V INSTRSUPAUTOTRANSFERSW DL	040136								
292	125V DC BATT CHARGER 1A IN&WIR	040147	3*	0*	3	0	0	0	2 FEB 1970	4 FEB 1970
185	NUCSERVICE&CONTROL HOUSE CONST	010199								
133	125V DC BATT CHARGER 1A&1B DL	040149								

ELECTRICAL INSTALLATION

```
                                              MAR 1969                APR 1969                 MAY 1969                JUN 1969
     ACTIVITY                   MEN  DAYS  0000011111111122222223000000011111112222222300000001111111222222222000000111111111222222 3
       NAME                     REQ SLACK  3456701234789014567811234789014567812345890125678923456901236789234569012367890345670

OFFICE&GUARD BLDG LIGHT CON IN   0     0   XXXXXXXX
34.5KV SWRK A CONSTRUCTION       0   864   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXC
230KV SWRK CONSTRUCTION          0   731   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXC
YARD CONDUIT&MANHOLE SYSTEM IN   0     0   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXC
34.5KV UG SWRK B CONSTRUCTION    0   824   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXC
OFFICE&GUARD BLDG LIGHT INGWIR   0     0           XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
SHOP&WREHSE B LIGHT CONDUIT IN   0     0                                     XXXXXXXXXXXXXXXXXXXXXX
FA&WDB UTILITY SERVICE FA IN     0   952                                         XXXXXXXXXXXXXXX
SHOP&WREHSE BLDG LIGHT INGWIR    0     0                                                         XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXC
TURB-GEN CABLE TRAY SYS IN       0     0                                                              XXXXXXXXXXXXXXXXXXXXXXXXXXC
TOTAL MANPOWER REQUIREMENTS                0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
                                           0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
                                           0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
```

ELECTRICAL INSTALLATION

```
                                              JUL 1969              AUG 1969             SEP 1969              OCT 1969
          ACTIVITY             MEN  DAYS 000000111111122222223300000011111112222222200000011111111222222300000001111112222222233
            NAME               REQ SLACK 123789014567812345890114567812345890125678923458901256789234569012367890345670123478901
TURB-GEN CABLE TRAY SYS IN       0     0 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
SHOP&WREHSE BLDG LIGHT IN&WIR    0     0 XXX
34.5KV UG SWRK B CONSTRUCTION    0   624 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXC
YARD CONDUIT&MANHOLE SYSTEM IN   0     0 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXC
230KV SWRK CONSTRUCTION          0   731 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXC
34.5KV SWRK A CONSTRUCTION       0   864 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXC
CABLE TUNNEL CABLE TRAY IN       0     0         XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
NSC BLDG CABLE TRAY SYS IN       0     0         XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
NSC BLDG LIGHTING CONDUIT IN     0     0              XXXXXXXXXX
CONTROLROOM LUMINOUS CEILIN&WI   0     0                        XXXXXXXXXXXXXXXXXXXXXX
DRY TYPE LIGHT&INSTR TR IN&WIR   0     0                        XXXXXXXXXXXXXXXXXXXXXX
NSC BLDG LIGHTING IN&WIR         0     0                        XXXXXXXXXXXXXXXXXXXXXX
NSC BLDG TERM RM CABLE TRAY IN   0     0                        XXXXXXXXXXXXXXXXXXXXXX
NSC BLDG CABLE RISER IN          0     0                                XXXXXXXXXX
480V LIGHT DISTR BD IN&WIR       0     0                                    XXXXXXXXXX
4KV SWITCHGEAR 1A IN&WIR         0     0                                          XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXC
4KV SWITCHGEAR 1B IN&WIR         0     0                                          XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXC
4KV SWITCHGEAR 1C IN&WIR         0     0                                          XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXC
4KV SWITCHGEAR 1D IN&WIR         0     0                                          XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXC
CONTROL BOARD IN&WIRING          0     0                                              XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXC
FWP FOUND GRDGRID DL             0  1070                                                                       X
FEEDWATER PMPS GRDGRID IN        0     0                                                                        XXX
CHEM FEED INSTR&CONTR EQ IN&WI   0     0                                                                                    XXXC
TOTAL MANPOWER REQUIREMENTS              000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
                                         000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
                                         000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
```

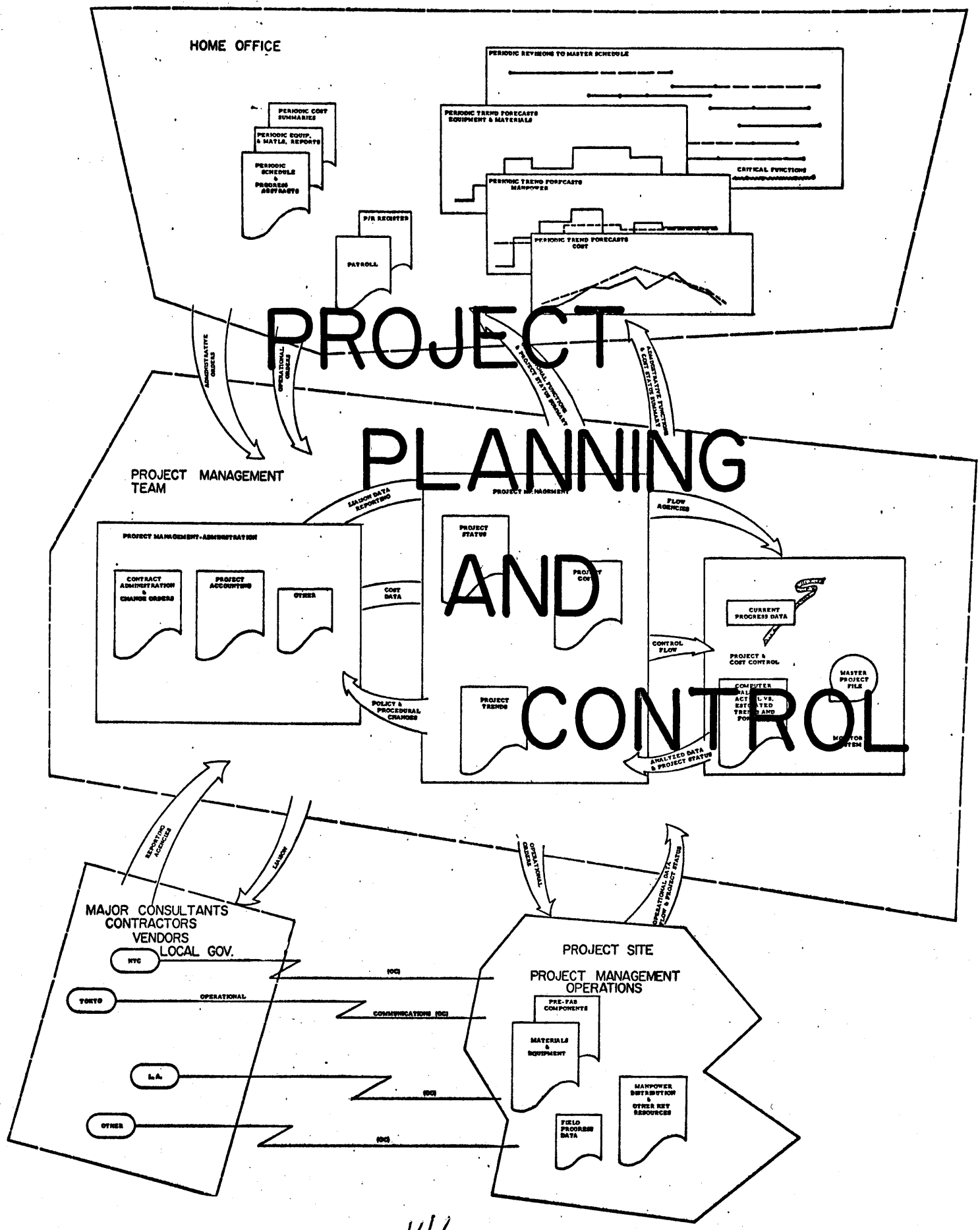
A18

145

OMNIMETRICS

PERT/CPM PANEL PRESENTATION BY:
Mr. Marvin Rubinstein and
Mr. Dave Stadlman

December 7, 1965



CRITICAL PATH METHOD/COSTING/GRAPHIC PLOT SYSTEM

The System

A unique computer-generated network diagram plot will be utilized to show the logic and interdependency of all scheduled jobs and their associated resources. (See Exhibit B and Exhibit F). The network diagram consists of each job labeled, time-scaled and dated for quick reference, providing an excellent base for cost and manpower planning, optimization and control.

Essentially, the system consists of four major subprograms:

1. A subprogram to generate a schedule based on a precise diagrammatic representation of a project. The diagramming is a simplified modification of the arrow and precedence network techniques.

2. A subprogram to evaluate available resources (labor, money, materials, etc.) against the above schedule. This process will partially optimize and revise the Critical Path previously calculated.

3. A subprogram to automatically interpret and output graphically in time scaled form, all elements of a project. The plotting is performed on an X-Y plotter controlled by computer programming. The end result, is a graphic document which can be used with little or no supplemental tabulations to evaluate progress, task inter-relationships, etc.

4. A fourth and optional subprogram is the job costing. This program was designed to allow for flexibility in cost coding and made part of a disk monitored system. Once the unique accounting function codes for a client are reduced to card records, large blocks of data may be easily processed. Detail of the generated reports can also be designed to meet the requirements of different management levels. (See Exhibit H for a typical client oriented variance report).

The System's Detailed Components

Time-scaling of the project schedule results in benefit to the planners and makes the schedule represented by the plot more easily visualized, so that it is more usable at all levels. The plot pictorially represents the complex interaction of each scheduled job and its required resources; i.e., skill and quantity of manpower, type and quantity of equipment or material. The plots eliminate the normal costly hunting through tabulations to find the effect of changes in the schedule. The effect due to schedule changes are readily seen, since jobs are "chained" in a logical sequence, determined by precedence. This technique provides the desired information in a form that the men working at the job site can understand and use easily and effectively. Updating and revisions are expeditiously processed with the system.

The network diagram plots (See Exhibit F) are supplemented with computer-generated detailed tabular information sorted in the desired sequences, (See Exhibit D) including the following for each job:

- Job Number
- Activity Description
- Node Identification
- Time Estimate
- Early Start Date
- Early Finish Date
- Late Start Date
- Late Finish Date
- Free Float Time
- Total Float Time
- Estimated Cost
- Resource Codes and Quantities
- Critical Path Indicator

The resources required plots (See Exhibit G) are also supplemented with computer-generated detailed tabular information, (See Exhibit E) and include the following:

- Project Time each job or activity starts
- Job Number
- Description
- Time Estimate
- Start and Finish Times
- Float Time
- Cost
- Quantity of Resource Available (Men, Machines, Material)
- Quantity of Resource Required for particular job
- Cumulative Quantity of Resource being utilized at any point in time
- Message if and when Resource Requirements exceed that available
- Items on the Critical Path are indicated

The System's Advantages

Planning and scheduling using CPM/COST/PLOT offers the following advantages:

1. Portrays graphically the work to be done in a manner which easily reveals the relationships between its separate parts. Also, it provides an easy means of communication between all functions of a business, such as estimating, purchasing, receiving, scheduling and construction.

2. For each job or activity, limits on the range of possible start dates gives the planner information for coordinating specific phases of the job with sub-contractors and material suppliers.

3. Reveals whether or not delays in doing specific activities will influence total job time.

4. Pinpoint the portions of the job that should be examined by the planner if he needs to reduce the expected job time. This is usually a substantial portion of the total number of activities and a total re-examination of the job is necessary. The graphical resource evaluation technique greatly reduces this time consuming phase of analysis.

5. Reveal to the planner instances where activities to be performed at the same time are in conflict or require the same resources. This type of situation may lead to replanning portions of the project.

6. Because the system is sufficiently flexible, the problem of relating project accounting functions to progress has been greatly simplified.

7. Allows scheduling and planning to be done in two distinct phases. This avoids duplication of effort and allows a more coherent approach to the problem.

8. By having an easy way to describe the job, (utilizing the computer-generated time-scaled network plot) examination of alternate methods and their impact on the total job has resulted in large savings in many instances.

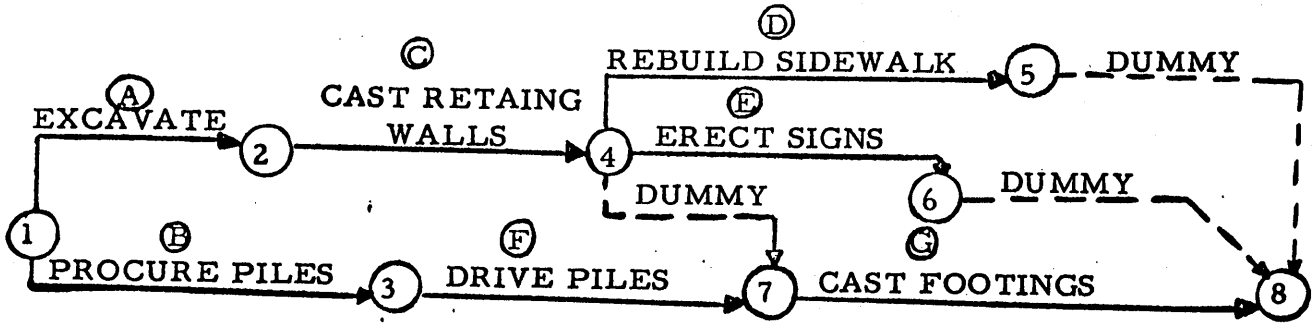
9. No restriction on the number of jobs or activities that are to be scheduled.

10. Any number of starting or ending jobs can be simultaneously indicated. The system is not restricted to just one starting and ending node like other CPM Programs.

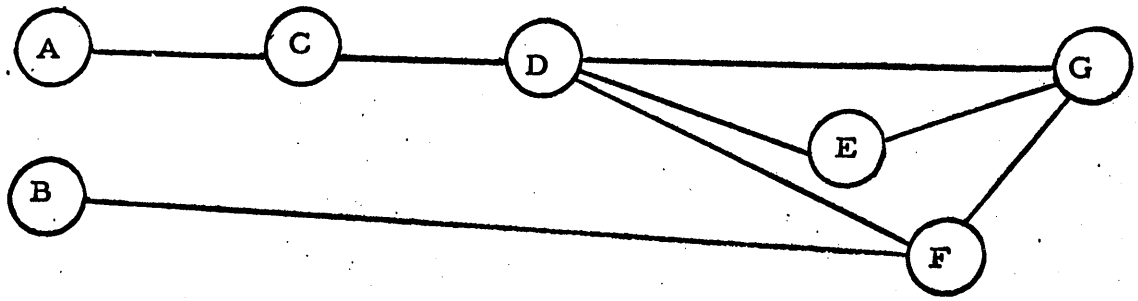
11. Jobs are identified by Job Number rather than Node Numbers. If a job is rescheduled, the Node Number may change; whereas, the Job Number will remain the same. This will provide for easy following of all scheduled jobs, regardless of schedule changes.

NETWORK DIAGRAMMING

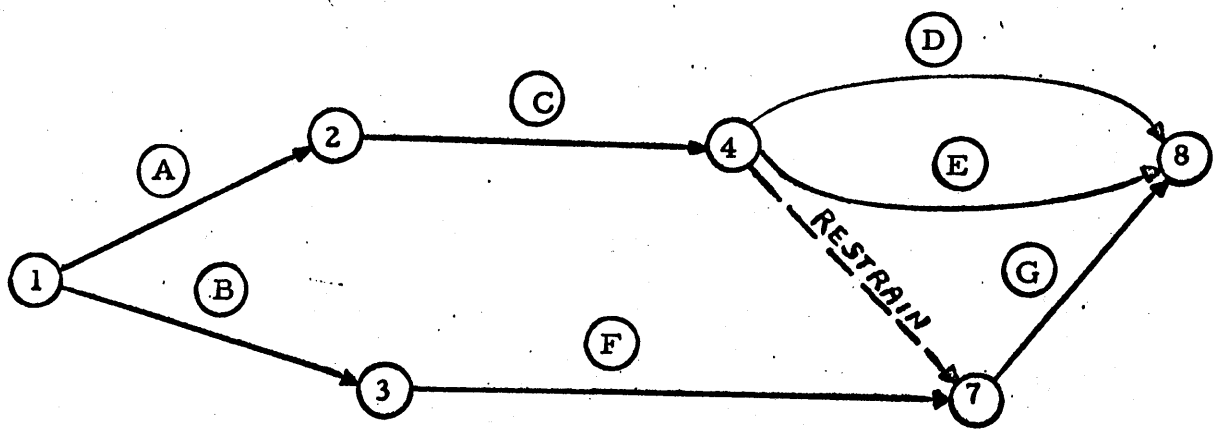
The Omnimetrics CPM/COST/PLOT System has been designed to handle all three variations of network diagramming:



Conventional Arrow Diagram

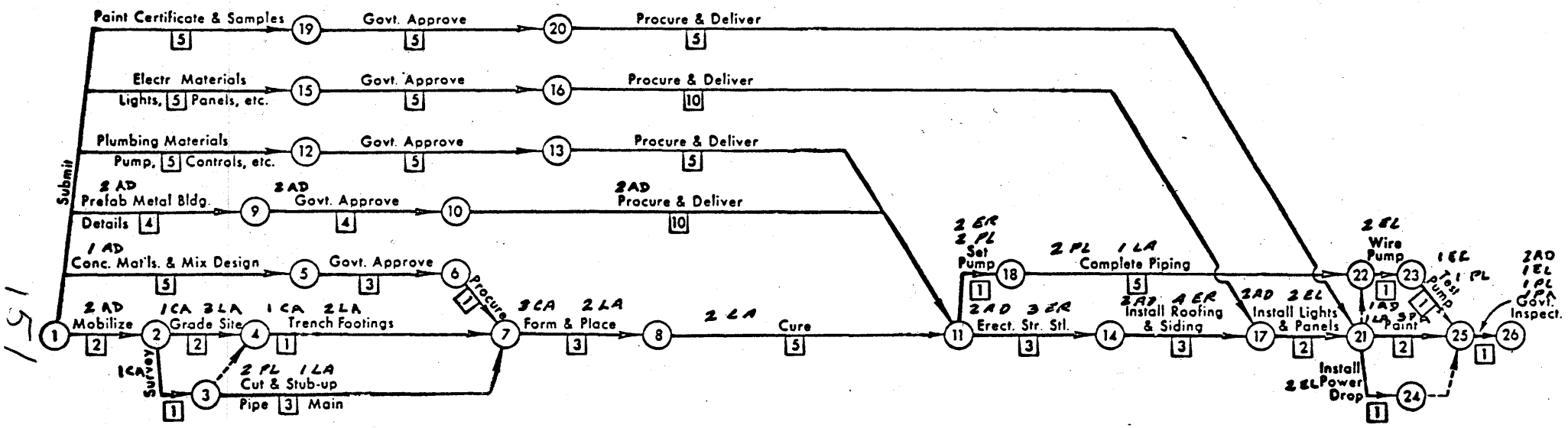


Precedence Diagram

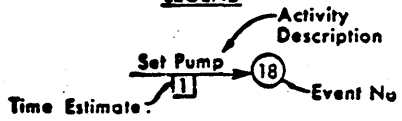


Modified Arrow Diagram

SAMPLE OF NETWORK PLAN FOR BOOSTER PUMP STATION



LEGEND



AD = Administration
ER = Erector
EL = Electrician
PL = Plumber
CA = Carpenter
LA = Laborer
PA = Painter

B

Date 8-1-65

Project Booster Pump Station

By Omnimetrics

CRITICAL PATH/COSTING/GRAPHIC PLOT SYSTEM

INPUT FORM

I	J	Time EST	JOB DESCRIPTION	Non-parallel Code	Est. Cost	Code/ Qnt.	Code/ Qnt.	Code/ Qnt.	Code/ Qnt.	Code/ Qnt.	SRT/ Cost Code	Job No.	D/B Code
1	19	5	SUBMIT PAINT CERT + SAMPLES									1	
1	15	5	SUBMIT ELECT MATLS									2	
1	12	5	SUBMIT PLBG MATLS									3	
1	9	4	SUBMIT PRE-FAB MET BLDGS D			1 2						4	
1	5	5	SUBMIT CONC MATLS + MIX DES			1 1						5	
1	2	2	MOBILIZE		2000	1 2						6	
2	4	2	GRADE SIZE		4000	5 1	6 3					7	
2	3	1	SURVEY		1000	5 1						8	
4	7	1	TRENCH FOOTINGS		1000	5 1	6 2					9	
3	4	0	RESTRAIN									10	
3	7	0	CUT + STUB-UP PIPE MAIN		3000	4 2	6					11	
5	6	3	GOVT OK									12	
6	7	1	PROCURE									13	
9	10	4	GOVT OK			1 2						14	
19	20	5	GOVT OK									15	
15	16	5	GOVT OK									16	
12	13	5	GOVT OK									17	
7	8	3	FORM + PLACE		6500	5 3	6 2					18	
8	11	5	CURE		500	6 2						19	
10	11	10	PROCURE + DELIVER		27000	1 2						20	
1	5	9	13	38	40	46	51	56	61	66	71	75	79

152

C-1

Date_____

Project_____

By_____

CRITICAL PATH/COSTING/GRAPHIC PLOT SYSTEM

INPUT FORM

I	J	Time EST	JOB DESCRIPTION	Non-parald Code	Est. Cost	Code/ Qnt.	Code/ Qnt.	Code/ Qnt.	Code/ Qnt.	Code/ Qnt.	SRT/ Cost Code	Job No.	D/B Code
13	11	5	PROCURE + DELIVER		21500							21	
16	17	10	PROCURE + DELIVER		4000							22	
20	21	5	PROCURE + DELIVER		2100							23	
11	18	1	SET PUMP		7000	2 2	4 2					24	
11	14	3	ERECT STR STL		3000	1 2	2 3					25	
18	22	5	COMPLETE PIPING		4000	4 2	6 1					26	
14	17	3	INSTALL ROOFING + SIDING		5000	1 2	2 4	6 1				27	
17	21	2	INSTALL LIGHTS + PANELS		2200	1 2	3 2					28	
22	23	1	WIRE PUMP		1500	3 2						29	
23	25	1	TEST PUMP		1200	3 1	4 1					30	
21	25	2	PAINT		5000	1 2	7 3	6 1				31	
21	25	1	INSTALL POWER DROP		1200	3 2						32	
25	26	2	GOVT INSPEC		700	1 2	3 1	4 1	7 1			33	
21	22	0	RESTRAIN									34	
1	5	9	13	38	40	46	51	56	61	66	71	75	79

193

C-2

NAP REPORT NO. 1 OMNIMETRICS CRITICAL PATH/COSTING/PLOTTING SYSTEM

CPM SCHEDULE SAMPLE SCHEDULE FOR BOOSTER PUMP STATION

* MARKS CRITICAL PATH AND RESOURCES REQUIRED ANALYSIS

SORTED BY ES,LF

	JOB NO.	ACTIVITY DESCRIPTION	START NODE	END NODE	TIME EST.	EARLY START	EARLY FINISH	LATE START	LATE FINISH	FREE FLOAT	TOTAL FLOAT	EST. COST	RESOURCES REQUIRED BY THIS ACTIVITY RESOURCE CODE/QUANTITY
*	4	SUBMIT PRE-FAB MET BLDG S	1	9	4	08/02/65	08/06/65	08/02/65	08/06/65	0	0	0	01/002
	5	SUBMIT CONC MATLS+MIX DES	1	5	5	08/02/65	08/09/65	08/03/65	08/10/65	0	1	0	01/001
	6	MOBILIZE	1	2	2	08/02/65	08/04/65	08/06/65	08/10/65	0	4	2000	01/002
	3	SUBMIT PLBG MATLS	1	12	5	08/02/65	08/09/65	08/05/65	08/12/65	0	3	0	
	2	SUBMIT ELECT MATLS	1	15	5	08/02/65	08/09/65	08/06/65	08/13/65	0	4	0	
	1	SUBMIT PAINT CERT+SAMPLES	1	19	5	08/02/65	08/09/65	08/17/65	08/24/65	0	11	0	
	8	SURVEY	2	3	1	08/04/65	08/05/65	08/10/65	08/11/65	0	4	1000	05/001
	7	GRADE SITE	2	4	2	08/04/65	08/06/65	08/11/65	08/13/65	0	5	4000	05/001 06/003
	10	RESTRAIN	3	4	0	08/05/65	08/05/65	08/13/65	08/13/65	1	6	0	
	11	CUT+STUB-UP PIPE MAIN	3	7	3	08/05/65	08/10/65	08/11/65	08/16/65	3	4	3000	04/002 06/001
*	14	GOVT OK	9	10	4	08/06/65	08/12/65	08/06/65	08/12/65	0	0	0	01/002
	9	TRENCH FOOTINGS	4	7	1	08/06/65	08/09/65	08/13/65	08/16/65	4	5	1000	05/001 06/002
	12	GOVT OK	5	6	3	08/09/65	08/12/65	08/10/65	08/13/65	0	1	0	
	17	GOVT OK	12	13	5	08/09/65	08/16/65	08/12/65	08/19/65	0	3	0	
	16	GOVT OK	15	16	5	08/09/65	08/16/65	08/13/65	08/20/65	0	4	0	
	15	GOVT OK	19	20	5	08/09/65	08/16/65	08/24/65	08/31/65	0	11	0	
	13	PROCURE	6	7	1	08/12/65	08/13/65	08/13/65	08/16/65	0	1	0	
*	20	PROCURE+DELIVER	10	11	10	08/12/65	08/26/65	08/12/65	08/26/65	0	0	27000	01/002
	18	FORM+PLACE	7	8	3	08/13/65	08/18/65	08/16/65	08/19/65	0	1	6500	05/003 06/002
	21	PROCURE+DELIVER	13	11	5	08/16/65	08/23/65	08/19/65	08/26/65	3	3	21500	
	22	PROCURE+DELIVER	16	17	10	08/16/65	08/30/65	08/20/65	09/03/65	4	4	4000	
	23	PROCURE+DELIVER	20	21	5	08/16/65	08/23/65	08/31/65	09/08/65	11	11	2100	
	19	CURE	8	11	5	08/18/65	08/25/65	08/19/65	08/26/65	1	1	500	06/002
	24	SET PUMP	11	18	1	08/26/65	08/27/65	08/30/65	08/31/65	0	2	7000	02/002 04/002
*	25	ERECT STR STL	11	14	3	08/26/65	08/31/65	08/26/65	08/31/65	0	0	3000	01/002 02/003
	26	COMPLETE PIPING	18	22	5	08/27/65	09/03/65	08/31/65	09/08/65	2	2	4000	04/002 06/001
*	27	INSTALL ROOFING+SIDING	14	17	3	08/31/65	09/03/65	08/31/65	09/03/65	0	0	5000	01/002 02/004 06/001
*	28	INSTALL LIGHTS+PANELS	17	21	2	09/03/65	09/08/65	09/03/65	09/08/65	0	0	2200	01/002 03/002
*	34	RESTRAIN	21	22	0	09/08/65	09/08/65	09/08/65	09/08/65	0	0	0	
*	29	WIRE PUMP	22	23	1	09/08/65	09/09/65	09/08/65	09/09/65	0	0	1500	03/002
*	31	PAINT	21	25	2	09/08/65	09/10/65	09/08/65	09/10/65	0	0	5000	01/002 07/003 06/001
	32	INSTALL POWER DROP	21	25	1	09/08/65	09/09/65	09/09/65	09/10/65	1	1	1200	03/002
*	30	TEST PUMP	23	25	1	09/09/65	09/10/65	09/09/65	09/10/65	0	0	1200	03/001 04/001
*	33	GOVT INSPEC	25	26	2	09/10/65	09/14/65	09/10/65	09/14/65	0	0	700	01/002 03/001 04/001 07/001

TOTAL ESTIMATED DURATION 30 COMPLETION DATE 09/14/65 TOTAL ESTIMATED COST 103,400

CODE	RESOURCE DESCRIPTION
1	ADMINISTRATION (AD)
2	ERECTORS (ER)
3	ELECTRICIANS (EL)
4	PLUMBERS (PL)
5	CARPENTERS (CA)
6	LABORERS (LA)
7	PAINTERS (PA)

157

FOR RESOURCE CODE 01 SAMPLE SCHEDULE FOR BOOSTER PUMP STATION

* MARKS CRITICAL PATH AND RESOURCES REQUIRED ANALYSIS

	PROJ TIME	JOB NO.	ACTIVITY DESCRIPTION	START NODE	END NODE	TIME EST.	EARLY START	EARLY FINISH	LATE START	LATE FINISH	FREE FLOAT	TOTAL FLOAT	EST. COST	TOTAL AVAIL	THIS JOB	CURR TOTAL
*	0	4	SUBMIT PRE-FAB MET BLDG S	1	9	4	0	4	0	4	0	0	0	2	2	2
	0	5	SUBMIT CONC MATLS+MIX DES	1	5	5	0	5	1	6	0	1	0	2	1	3
	0	6	MOBILIZE	1	2	2	0	2	4	6	0	4	2000	2	2	5
	2	6	COMPLETED											2	2	3
	4	4	COMPLETED											2	2	1
*	4	14	GOVT OK	9	10	4	4	8	4	8	0	0	0	2	2	3
	5	5	COMPLETED											2	1	2
	8	14	COMPLETED											2	2	0
*	8	20	PROCURE+DELIVER	10	11	10	8	18	8	18	0	0	27000	2	2	2
	18	20	COMPLETED											2	2	0
*	18	25	ERECT STR STL	11	14	3	18	21	18	21	0	0	3000	2	2	2
	21	25	COMPLETED											2	2	0
*	21	27	INSTALL ROOFING+SIDING	14	17	3	21	24	21	24	0	0	5000	2	2	2
	24	27	COMPLETED											2	2	0
*	24	28	INSTALL LIGHTS+PANELS	17	21	2	24	26	24	26	0	0	2200	2	2	2
	26	28	COMPLETED											2	2	0
*	26	31	PAINT	21	25	2	26	28	26	28	0	0	5000	2	2	2
	28	31	COMPLETED											2	2	0
*	28	33	GOVT INSPEC	25	26	2	28	30	28	30	0	0	700	2	2	2

155

REQUIREMENT REPORT — OMNIMETRICS CRITICAL PATH/COSTING/PLOTTING SYSTEM

FOR RESOURCE CODE 02 — SAMPLE SCHEDULE FOR BOOSTER PUMP STATION

* MARKS CRITICAL PATH — AND RESOURCES REQUIRED ANALYSIS

	PROJ TIME	JOB NO.	ACTIVITY DESCRIPTION	START NODE	END NODE	TIME EST.	EARLY START	EARLY FINISH	LATE START	LATE FINISH	FREE FLOAT	TOTAL FLOAT	EST. COST	TOTAL AVAIL	THIS JOB	CURR TOTAL
	18	24	SET PUMP	11	18	1	18	19	20	21	0	2	7000	4	2	2
*	18	25	ERECT STR STL	11	14	3	18	21	18	21	0	0	3000	4	3	5
	19	24	COMPLETED											4	2	3
	21	25	COMPLETED											4	3	0
*	21	27	INSTALL ROOFING+SIDING	14	17	3	21	24	21	24	0	0	5000	4	4	4
	24	27	COMPLETED											4	4	0

FOR RESOURCE CODE 03 SAMPLE SCHEDULE FOR BOOSTER PUMP STATION

* MARKS CRITICAL PATH AND RESOURCES REQUIRED ANALYSIS

	PROJ TIME	JOB NO.	ACTIVITY DESCRIPTION	START NODE	END NODE	TIME EST.	EARLY START	EARLY FINISH	LATE START	LATE FINISH	FREE FLOAT	TOTAL FLOAT	EST. COST	TOTAL AVAIL	THIS JOB	CURR TOTAL
*	24	28	INSTALL LIGHTS+PANELS	17	21	2	24	26	24	26	0	0	2200	2	2	2
	26	28	COMPLETED											2	2	0
*	26	29	WIRE PUMP	22	23	1	26	27	26	27	0	0	1500	2	2	2
	26	32	INSTALL POWER DROP	21	25	1	26	27	27	28	1	1	1200	2	2	4
	27	29	COMPLETED											2	2	0
	27	32	COMPLETED											2	2	0
*	27	30	TEST PUMP	23	25	1	27	28	27	28	0	0	1200	2	1	1
	28	30	COMPLETED											2	1	0
*	28	33	GOVT INSPEC	25	26	2	28	30	28	30	0	0	700	2	1	1

157

REQUIREMENT REPORT

FOR RESOURCE CODE 04

* MARKS CRITICAL PATH

OMNIMETRICS CRITICAL PATH/COSTING/PLOTTING SYSTEM

SAMPLE SCHEDULE FOR BOOSTER PUMP STATION

AND RESOURCES REQUIRED ANALYSIS

	PROJ TIME	JOB NO.	ACTIVITY DESCRIPTION	START NODE	END NODE	TIME EST.	EARLY START	EARLY FINISH	LATE START	LATE FINISH	FREE FLOAT	TOTAL FLOAT	EST. COST	TOTAL AVAIL	THIS JOB	CURR TOTAL
	3	11	CUT+STUB-UP PIPE MAIN	3	7	3	3	6	7	10	3	4	3000	2	2	2
	6	11	COMPLETED											2	2	0
	18	24	SET PUMP	11	18	1	18	19	20	21	0	2	7000	2	2	2
	19	24	COMPLETED											2	2	0
	19	26	COMPLETE PIPING	18	22	5	19	24	21	26	2	2	4000	2	2	2
	24	26	COMPLETED											2	2	0
*	27	30	TEST PUMP	23	25	1	27	28	27	28	0	0	1200	2	1	1
	28	30	COMPLETED											2	1	0
*	28	33	GOVT INSPEC	25	26	2	28	30	28	30	0	0	700	2	1	1

FOR RESOURCE CODE 05 SAMPLE SCHEDULE FOR BOOSTER PUMP STATION

* MARKS CRITICAL PATH AND RESOURCES REQUIRED ANALYSIS

PROJ TIME	JOB NO.	ACTIVITY DESCRIPTION	START NODE	END NODE	TIME EST.	EARLY START	EARLY FINISH	LATE START	LATE FINISH	FREE FLOAT	TOTAL FLOAT	EST. COST	TOTAL AVAIL	THIS JOB	CURR TOTAL
2	8	SURVEY	2	3	1	2	3	6	7	0	4	1000	2	1	1
2	7	GRADE SITE	2	4	2	2	4	7	9	0	5	4000	2	1	2
3	8	COMPLETED											2	1	1
4	7	COMPLETED											2	1	0
4	9	TRENCH FOOTINGS	4	7	1	4	5	9	10	4	5	1000	2	1	1
5	9	COMPLETED											2	1	0
9	18	FORM+PLACE	7	8	3	9	12	10	13	0	1	6500	2	3	3
9	18	CANNOT BE RUN - TOTAL REQUIREMENT GREATER THAN AVAILIBILITY													
12	18	COMPLETED											2	3	0

159

REQUIREMENT REPORT — OMNIMETRICS CRITICAL PATH/COSTING/PLOTTING SYSTEM — PAGE 6

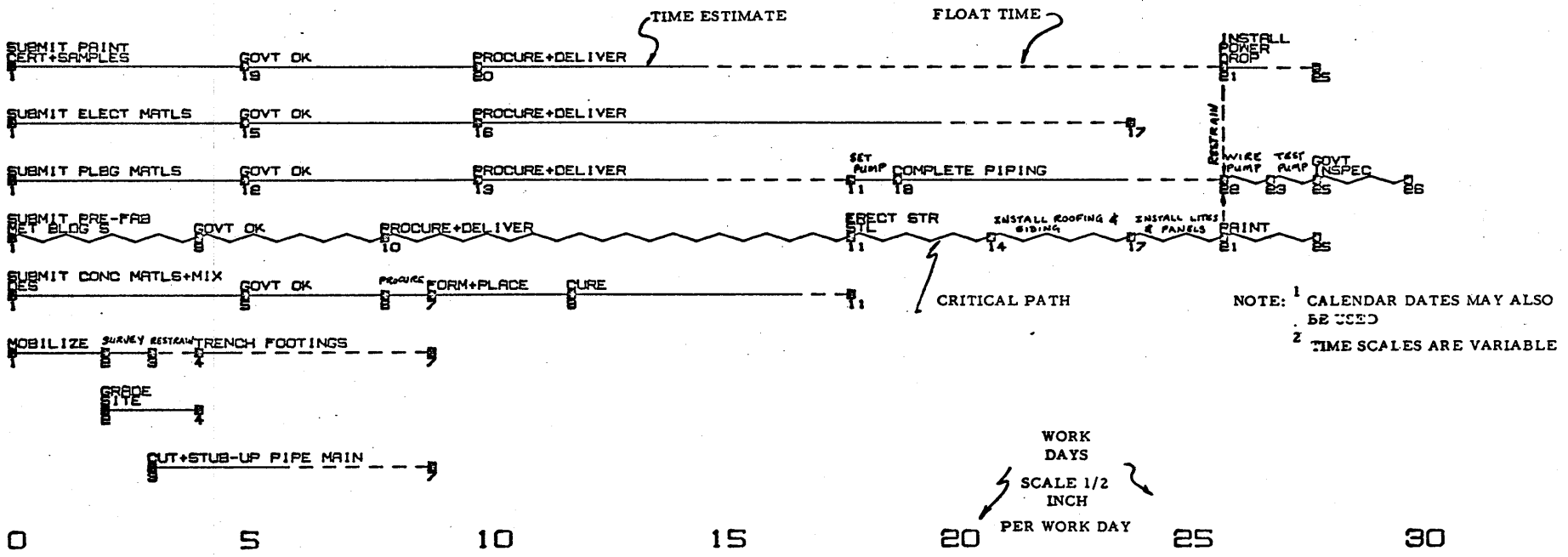
FOR RESOURCE CODE 06 — SAMPLE SCHEDULE FOR BOOSTER PUMP STATION

* MARKS CRITICAL PATH — AND RESOURCES REQUIRED ANALYSIS

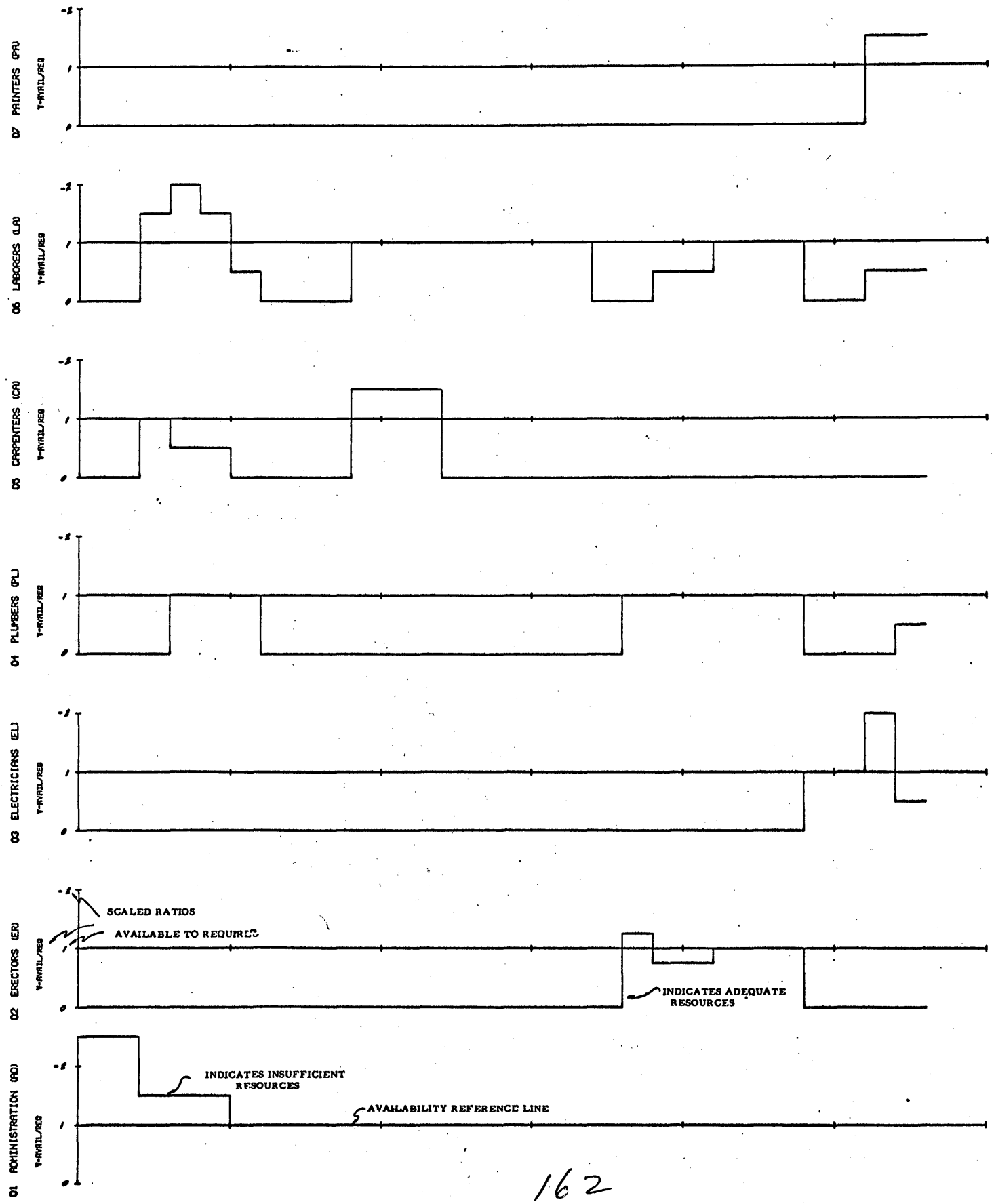
	PROJ TIME	JOB NO.	ACTIVITY DESCRIPTION	START NODE	END NODE	TIME EST.	EARLY START	EARLY FINISH	LATE START	LATE FINISH	FREE FLOAT	TOTAL FLOAT	EST. COST	TOTAL AVAIL	THIS JOB	CURR TOTAL
	2	7	GRADE SITE	2	4	2	2	4	7	9	0	5	4000	2	3	3
	2	7	CANNOT BE RUN - TOTAL REQUIREMENT GREATER THAN AVAILIBILITY													
	3	11	CUT+STUB-UP PIPE MAIN	3	7	3	3	6	7	10	3	4	3000	2	1	4
	4	7	COMPLETED											2	3	1
	4	9	TRENCH FOOTINGS	4	7	1	4	5	9	10	4	5	1000	2	2	3
	5	9	COMPLETED											2	2	1
	6	11	COMPLETED											2	1	0
	9	18	FORM+PLACE	7	8	3	9	12	10	13	0	1	6500	2	2	2
	12	18	COMPLETED											2	2	0
	12	19	CURE	8	11	5	12	17	13	18	1	1	500	2	2	2
	17	19	COMPLETED											2	2	0
	19	26	COMPLETE PIPING	18	22	5	19	24	21	26	2	2	4000	2	1	1
*	21	27	INSTALL ROOFING+SIDING	14	17	3	21	24	21	24	0	0	5000	2	1	2
	24	26	COMPLETED											2	1	0
	24	27	COMPLETED											2	1	0
*	26	31	PAINT	21	25	2	26	28	26	28	0	0	5000	2	1	1
	28	31	COMPLETED											2	1	0

091

SAMPLE OF A TIME SCALED PLOT FOR A BOOSTER PUMP STATION



MANPOWER REQUIREMENTS ANALYSIS



PROJECT NO. 1100 VARIANCE REPORT FOR PERIOD ENDING 07/12/65 CURRENT COMPLETION 10/31/65 PAGE 001

JOB NO.	DESCRIPTION	EST. QUANT.	UNIT	PROGRESS THIS PERIOD QUANTITIES THIS PERIOD	TO DATE	O/O CMP	COST THIS PERIOD	UNIT COSTS ESTIMATE	THIS PERIOD	TO DATE	COSTS TO DATE EST.	ACT.	VAR.	TOTAL TO COMPLETE EST.	PRED.	VAR.	D O T E
	CATAGORY 001 JOB OVERHEAD																
001	SUPERVISION	40	WKS	2	24	60	219.09	200.0000	109.5450	162.1400	4800	3891	909	6000	6486	1514	N*
002	FIELD OFFICE	40	WKS	2	24	60	.00	8.7500	.0000	10.5833	210	254	(44)	350	423	(73)	N*
003	HAULING	40	WKS	2	24	60	17.65	62.5000	8.8250	15.2083	1500	365	1135	2500	608	1892	N*
	TOTALS FOR CATAGORY 001						236.74				6510	4510	2000	10850	7517	3333	
	CATAGORY 002 CONCRETE WORK																
	TASK 007 TILTUP WALLS AND COLUMN JOINERY (INCL. EDISON VAULT)																
081	FORMWORK - ADD 1/2 BEAM	17	LF	0	17	100	.00	1.5000	.0000	.0000	26	0	26	26	0	26	Y*
082	ANCHOR BOLTS FOR UPPER COLS	11	SETS	0	11	100	.00	3.0000	.0000	3.0000	33	33	0	33	33	0	Y
083	ANCHOR BOLTS FOR UPPER BEAMS	18	SETS	0	18	100	.00	2.0000	.0000	1.8700	36	34	2	36	34	2	Y
084	PLACE CONCRETE (3000LBS)	44	CY	0	46	104	.00	3.5000	.0000	1.5800	152	73	79	152	73	79	Y*
085	GROUT BASE PLATES	29	EA	0	25	86	.00	2.5000	.0000	1.4530	63	36	27	73	42	31	N*
086	SACK AND PATCH WALLS TO	9300	SF	0	6000	64	.00	.0300	.0000	.0326	180	196	(16)	279	315	(36)	N*
087	BLOCKOUTS FOR STEPPED FOOT.	5	EA	0	5	100	.00	7.0000	.0000	.0000	35	0	35	35	0	35	Y*
	TOTALS FOR TASK 007						.00				525	372	153	634	497	137	
	TASK 009 FIRST FLOOR SLAB INCL. EDISON VAULT ROOF																
095	SCREEDS	30783	SF	24000	71500	232	171.05	.0200	.0071	.0105	616	750	(134)	616	750	(134)	Y*
096	FORMWORK - DEPRESSED EDGES	425	LF	150	470	110	109.65	.3000	.7350	.2980	128	140	(12)	128	140	(12)	Y
097	FORMWORK - BEAM - SOUTH	154	LF	20	154	100	131.69	1.7000	6.6100	2.7440	262	423	(161)	262	423	(161)	Y*
098	FORMWORK - NORTH AND EAST	115	LF	150	260	226	.00	1.5000	.0000	0.6080	173	158	15	173	158	15	Y
099	FORMWORK - BEAM AND CURB	335	LF	100	280	81	57.67	1.2500	.5800	2.7890	350	780	(431)	419	935	(516)	N*
100	FORMWORK - FLAT SUFFITS, ENT.	1458	LF	200	1358	93	190.81	.2500	.9550	0.3180	340	431	(91)	365	463	(98)	N*
107	PLACE CONCRETE	329	CY	136	354	108	277.64	1.8500	2.0400	2.0200	610	715	(105)	610	715	(105)	Y*
108	FINISH AND CURE	30783	SF	12400	32800	106	323.13	.0450	.0261	.0350	1385	1150	235	1385	1150	235	N*
109	RUNWAYS	32800	SF	26800	32800	100	20.40	.0100	.0075	.0070	328	231	97	328	231	97	Y
110	ANCHOR BOLTS	0	LS	0	0	0	.00	.0000	.0000	.0000	0	0	0	0	0	0	N
	TOTALS FOR TASK 009						1282.04				4192	4778	(587)	4286	4965	(679)	
	TOTALS FOR CATAGORY 002						1282.04				4717	5150	(434)	4920	5462	(542)	

TYPICAL PROJECT COSTING CONTROL REPORT

H

SOLUTION OF A PROBLEM IN HEAT TRANSFER

J. C. Caslin, H. E. Fettis
and
J. W. Goresh

In studying the effect of varying wall conditions on heat conduction in a fully developed turbulent gas flow in a long tube, it is necessary to solve the partial differential equation

$$\frac{\partial}{\partial x}\left\{(1-x^7)\ \frac{\partial \theta}{\partial x}\right\} = \lambda x^7\ \frac{\partial \theta}{\partial z} \tag{1}$$

The equation must be solved subject to the conditions

$$\left.\frac{\partial \theta}{\partial x} + k\theta = 0\right] \quad x = 0 \tag{2}$$

and θ at x= 1 be finite. Here λ is a physical parameter depending on the physical properties of the gas, θ, a non-dimensional temperature difference, k, a parameter depending on the conductivities of the gas and wall insulation, and r and z are non-dimensional radial and axial distances and $x = (1-r^2)^{1/7}$. In addition, the distribution of temperature at z= 0 is given as:

$$\theta(0, x) = f(x) \tag{3}$$

The complete solution of equation (1) is approximated as a sum of the series

$$\theta = \sum_{n=1}^{n} A_n e^{-a_n z}\ \psi_n(x) \tag{4}$$

where the constants A_n are chosen such that

$$\int_0^1 \left\{\sum \left[A_n\ \psi_n(x)\right] - f(x)\right\}^2\ x^7\ dx \tag{5}$$

is a minimum. The ψ_n are eigenfunctions of the ordinary differential equation

$$\frac{d}{dx}\left\{(1-x^7)\ \frac{d\psi_n}{dx}\right\} + \alpha_n \lambda x^7\ \psi_n = 0 \qquad (6)$$

These and the corresponding eigenvalues $\lambda\alpha_n$ are found by the Ritz method in terms of the eigen functions of the equation

$$\frac{d}{dx}\left\{(1-x^7)\ \frac{dy}{dx}\right\} + \beta x^5\ y = 0 \qquad (7)$$

which may be solved in terms of hypergeometric functions.

The program consists of a simple input-output routine for introducing the physical parameters. In order to find the eigen values and eigenfunctions for equation 7, it is necessary to solve a transcendental equation involving gamma functions. This is accomplished by a root-finding subroutine using "False Position" together with a subroutine for generating the gamma functions. These functions depend only on the parameter k. The eigenfunctions thus determined are then used as coordinate functions in a Ritz type analysis to obtain approximations to the eigenvalues and eigenfunctions of equation (6). This portion requires a subroutine which evaluates integrals involving products of hypergeometric series and a matrix eigenvalue-eigenvector routine based on the Jacobi diagonalization method. Finally, the initial temperature distribution is expressed in terms of the eigenfunctions of equation (6), making use of the orthogonality relations for these functions (according to equations (3) and (4)). With the A_n known, the temperature at any value of r and z may now be determined.

SOME APPLICATIONS OF CRACOVIANS

by

Marco Tulio Rincón B.

Geodetic Engineer

Proffessor of La Universidad del Zulia

Maracaibo - Venezuela

COMMON WESTERN REGION WINTER MEETING

December 6 - 7 - 8 1965

Los Angeles. California

166

CONTENT

PREFACE.

Arrival of calculating machines originated a revolution in numerical calculus and now a days, electronic computers represent the highest exponent of this marvelous revolution.-

The old classical logarithmic treatment of problems, often with long and tedious transformations only to make possible the use of logarithms, has been substituted by new calculating methods which still are in a continuous transformation and evolution, based on new concepts derived from existence of calculating machines.-

"Cracovian methods" constitute a good example of these new concepts. At first they were created because of the needs of new methods to compute with calculating machines but very soon, they showed a great analitical power which caused the origin of "Cracovian Algebra" by means of which it has been possible to attack a lot of problems with many advantages over classical methods.-

Some typical applications of Cracovian Algebra are:

Direct solution of problems in spherical poligonometry very common in Astronomy. The new method is shorter and simpler than classical method of decomposition into spherical triangles.-

Solution of linear equations systems by means of a typical cracovian operation which makes possible to solve practically systems with any number of unknowns.-

New treatment of Errors Theory and Least Squares Adjustment with very general procedures to solve symmetrical systems.-

In all these applications, cracovian calculus offers the following advantages:

a) A great analitical power verified by discovering of laws and relations in Physics and Mathematics Sciences.-

b) Brightness and briefness of cracovian expressions.-

c) Calculating procedures very easy to memorize, lightening in this way mental effort of calculator.-

d) Economy of calculations and good propagation of errors because of compact manner of cracovian calculating schemes which, in a fluid way and using only additions of products, make possible to shorten the intermediate notes.-

In spite of the mencionated advantages, cracovian methods are practically unknown. The capital reason for this is that the bibliography is written in polish, because their inventor, Dr. Tadeusz Banachiewicz, Director of the Astronomical Observatory in Cracow Poland, and his follower, like Kochsmanski, Sierpinski and Hausbrandt, are all polish.

However, in Venezuela and thanks to the interest of polish Prof. Ing. Bernardo Wahl, two articles written in Spanish have been published: "Cálculo de Cracovianos" (Cracovian calculus) and "La Aplicación del Algebra Cracoviana en el Cálculo de Compensación" (Application of Cracovian Algebra in Least Squares Adjustment). These two articles, published by the Engineering School of La Universidad del Zulia, - constitute the basis of this work.-

Two programs developed to solve linear equations systems by means of cracovian calculus, are described in this work. The first one is applicable to general linear systems; the second one is a modification of a program developed by the author in his special work to obtain the title of Engineer and is applicable only to normal equations systems derived from least squares adjustments. The last section of this - second program is devoted to calculate the mean square error of - unknowns although northamerican geodetist do not consider important this calculation. However, if it is possible to use an electronic - computer, it is advisable to calculate these errors which offer a very good criterion of measurement precission. Moreover, cracovian methods make possible this calculation with minnimum programming effort.

All these programs have been made with aids of an IBM 1620 with 1622 Card Read-Punch which works in the "Instituto de Cálculo Aplicado" in Engineering School.-

In addition to the descriptions of the programs, there are two sections about general definitions and basic operations in cracovian algebra in order to popularize these methods which, by the explained reasons, are practically unknown.- As it is explained in those sections, "cracovians" are rectangular arrays of numbers which obey to certain operations rules and whose name is derived from Cracow, natal city of Dr. Banachiewicz. There is a great similitude between matrix - algebra and cracovian algebra; nevertheless, there is a remarkable difference between them derived from the different way to accomplish multiplication: with matrices, this operation is rows by columns -

while with cracovians it is columns by columns. This fact, apparently with no importance, is the reason for all advantages of cracovian - algebra in calculations practice as it can be noticed in the applications described in the present work which are only a very little example of the innumerable cracovian applications.-

To finish these short lines, the author wishes to thank very - much the organizers of this meeting the opportunity to participate in such an active way and to express his hopes of having contributed to the divulgation of these calculation methods.-

Marco T. Rincón B.

-November, 1965.

1.- GENERAL DEFINITIONS [1]

1.1. "Cracovians" are rectangular arrays of numbers subject to the rules of operations given below.-

They will be represented by means of the symbols

$$\underline{a} = \begin{Bmatrix} a_{11} & a_{21} & \cdots & a_{t1} \\ a_{12} & a_{22} & \cdots & a_{t2} \\ \cdots & \cdots & \cdots & \cdots \\ a_{1n} & a_{2n} & \cdots & a_{tn} \end{Bmatrix} \quad ; \quad \underline{b} = \begin{Bmatrix} b_{11} & b_{21} & \cdots & b_{c1} \\ b_{12} & b_{22} & \cdots & b_{c2} \\ \cdots & \cdots & \cdots & \cdots \\ b_{1n} & b_{2n} & \cdots & b_{cn} \end{Bmatrix}$$

The numbers a_{rs} are called "elements of the cracovian". The first subscript indicates the column and the second subscript indicates the row in which the element stands.-

In agreement with the "Cracovian Calculus Norms In Geodetic - Computations" dated the 30 - 6 - 1959 and prepared by the "Polish Committee of Norms", the symbolical representations of cracovians are

a) For printing, black letters

b) In writing by hand or typewriter, underlined letters

1.2. A cracovian of m columns and n rows is said to be of order "m by n" or "m.n" .-

(1) This section and section 2 have been taken, with some modifications, from the article of Proffessors B. Wahl and Ernesto H. Battistella "Cálculo de Cracovianos" published in N° 2 of the "Revista de la Facultad de Ingeniería" - Universidad del Zulia - Maracaibo - Venezuela.

1.3. A "square cracovian" has equal number of columns and rows. It is said to be "diagonal" when $d_{rs}=0$ if $r \neq s$

$$\underline{d}=\begin{Bmatrix} d_{11} & 0 & \dots & 0 \\ 0 & d_{22} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & d_{nn} \end{Bmatrix}$$

The cracovian γ, or "unit cracovian" is a special case of diagonal cracovian in which $d_{rr}= 1$

$$\gamma=\begin{Bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{Bmatrix}$$

1.4. The elements $a_{11}, a_{22}, \dots, a_{nn}$ are called "diagonal elements". The straight line which contains all these elements is called "principal diagonal" and is a geometrical resemblance that helps to understand many operations of cracovian algebra.-

1.5. A cracovian, every element of which is zero, is called a "zero cracovian".-

1.6. The cracovian of order n.m obtained by interchanging the columns and rows of an m.n cracovian $\underline{a}$ is called the "transpose of $\underline{a}$ " and is denoted $\gamma\underline{a}$.-

1.7. A square cracovian is "symmetric" when it is equal to its transpose.-

2. BASIC OPERATIONS IN CRACOVIAN ALGEBRA.

2.1. Two cracovians are said to be equal if and only if each element of one is equal to the corresponding element of the other, that is

$$\underline{a} = \underline{b}$$

if and only if $a_{rs} = b_{rs}$

2.2 Addition or substraction of cracovians is accomplished by adding or substracting the corresponding elements. If

$$\underline{a} \pm \underline{b} = \underline{c}$$

then $a_{rs} \pm b_{rs} = c_{rs}$

This operation is only possible with cracovians of the same order.-

2.3. If "x" is a constant,

$$x \cdot \underline{a} = \underline{a} \cdot x = \underline{A}$$

is obtained by multiplying each element of $\underline{a}$ by "x", that is

$$A_{rs} = x \cdot a_{rs}$$

2.4. The product of a column of a cracovian by a column of other cracovian, with the same number of rows, is obtained by multi -

plying the elements of the same row and adding this binary products. Thus, the product of the r-th column of $\underline{a}$, named a_r, by the s-th column of $\underline{b}$, named b_s , is obtained in the following way

$$a_r \cdot b_s = \begin{Bmatrix} a_{r1} \\ a_{r2} \\ \cdots \\ a_{rn} \end{Bmatrix} \begin{Bmatrix} b_{s1} \\ b_{s2} \\ \cdots \\ b_{sn} \end{Bmatrix} = a_{r1} \cdot b_{s1} + a_{r2} \cdot b_{s2} + \cdots + a_{rn} \cdot b_{sn}$$

2.5. The product of a row of a cracovian by a row of other cracovian, with the same number of columns, is obtained by multiplying the elements of the same column and adding this binary products.-

2.6. The product $\underline{c}$ of a cracovian $\underline{a}$ by other cracovian $\underline{b}$, only - possible when they have the same number of rows, is obtained - from

$$\underline{a} \cdot \underline{b} = \begin{Bmatrix} a_{11} & a_{21} & \cdots & a_{t1} \\ a_{12} & a_{22} & \cdots & a_{t2} \\ \cdots & \cdots & \cdots & \cdots \\ a_{1n} & a_{2n} & \cdots & a_{tn} \end{Bmatrix} \cdot \begin{Bmatrix} b_{11} & b_{21} & \cdots & b_{e1} \\ b_{12} & b_{22} & \cdots & b_{e2} \\ \cdots & \cdots & \cdots & \cdots \\ b_{1n} & b_{2n} & \cdots & b_{en} \end{Bmatrix} = \begin{Bmatrix} c_{11} & c_{21} & \cdots & c_{t1} \\ c_{12} & c_{22} & \cdots & c_{t2} \\ \cdots & \cdots & \cdots & \cdots \\ c_{1e} & c_{2e} & \cdots & c_{te} \end{Bmatrix} = \underline{c}$$

in which $c_{rs} = a_{r1} \cdot b_{s1} + a_{r2} \cdot b_{s2} + \cdots + a_{rn} \cdot b_{sn}$

The elements of $\underline{c}$ are obtained by multiplying the columns of $\underline{a}$ by the columns of $\underline{b}$. The multiplication in this way, so different from matrices multiplication, is the most important difference between the algebras of the two groups and the capital reason for a lot of advantages of cracovian algebra in calculations practice.-

2.7. Cracovian multiplication is neither commutative nor associative, that is

$$\underline{a} \cdot \underline{b} \neq \underline{b} \cdot \underline{a}$$
$$(\underline{a} \cdot \underline{b}) \cdot \underline{c} \neq \underline{a} \cdot (\underline{b} \cdot \underline{c})$$

However,

$$\underline{a} \cdot \underline{b} = \tau(\underline{b} \cdot \underline{a})$$
$$(\underline{a} \cdot \underline{b}) \cdot \underline{c} = \underline{a} \cdot (\underline{c} \cdot \tau\underline{b})$$

are valid relations.-

2.8. The square of a cracovian $\underline{a}^2$ is the product of $\underline{a} \cdot \underline{a}$ and is always a square symmetric cracovian.-

2.9. A cracovian quotient is the result of cracovians division

$$\underline{c} = \underline{a} : \underline{b}$$

This operation may be substituted by the product

$$\underline{c} \cdot \tau\underline{b} = \underline{a}$$

2.10. Any cracovian different from zero may be decomposed into two "canonical cracovians". In a canonical cracovian all the elements under the principal diagonal are equal to zero, thus

$$a_{rs} = 0 \quad \text{if} \quad r < s$$

The number of possible solutions is reduced by stablishing to the decomposition

$$\underline{a} = \underline{g} \cdot \underline{h}$$

that all the diagonal elements of $\underline{g}$ be equal to 1 , that is, in the case of a square cracovian of order 3

$$\begin{Bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{Bmatrix} = \begin{Bmatrix} 1 & g_{21} & g_{31} \\ 0 & 1 & g_{32} \\ 0 & 0 & 1 \end{Bmatrix} \cdot \begin{Bmatrix} h_{11} & h_{21} & h_{31} \\ 0 & h_{22} & h_{32} \\ 0 & 0 & h_{33} \end{Bmatrix}$$

It can be noticed that the canonical cracovians of a square cracovian, are square too.-

2.11. The "cracovian root" of a symmetric cracovian is obtained by the decomposition into two equal canonical cracovians. This can be written in the following way

$$(\sqrt{\underline{a}})^2 = \underline{a}$$

In geodetic problems, it is always possible to obtain the cracovian root as explained later.-

2.12. The inverse of a cracovian $\underline{a}$ designated $\underline{a}^{-1}$ is a cracovian whose product by the cracovian $\underline{a}$ is the unit cracovian, that is

$$\underline{a} \cdot \underline{a}^{-1} = \gamma$$

and also

$$\underline{a}^{-1} \cdot \underline{a} = \gamma$$

The same is valid for the traspose

$$\gamma \underline{a} \cdot \gamma \underline{a}^{-1} = \gamma$$
$$\gamma \underline{a}^{-1} \cdot \gamma \underline{a} = \gamma$$

2.13. The calculation of the inverse of a cracovian is only possible when its determinant is different from zero. Once this condition is fulfiled this calculation may be accomplished in different ways

a) Using the same procedure to calculate the inverse of a matrix.

b) Calculating the inverse of the canonical cracovians into which it can be decomposed, since if

$$\underline{a} = \underline{g} \cdot \underline{h}$$

then

$$\underline{a}^{-1} = \underline{g}^{-1} \cdot \underline{h}^{-1}$$

It is very easy to calculate the inverse of a canonical cracovian because

$$\begin{Bmatrix} h_{11} & h_{21} & h_{31} \\ 0 & h_{22} & h_{32} \\ 0 & 0 & h_{33} \end{Bmatrix}^{-1} = \begin{Bmatrix} h_{11}^{-1} & 0 & 0 \\ x_{12} & h_{22}^{-1} & 0 \\ x_{13} & x_{23} & h_{33}^{-1} \end{Bmatrix}$$

and the unknowns are obtained from the definition $\underline{h}.\underline{h}^{-1} = \gamma$.-

c) If $\underline{a}$ is a symmetric cracovian, then $\underline{g} = \underline{h} = \sqrt{\underline{a}}$ and

$$\underline{a}^{-1} = \left[(\sqrt{\underline{a}})^{-1} \right]^2$$

Thus, the inverse of a symmetric cracovian can be easily obtained from the inverse of its cracovian root.-

3. GENERAL SOLUTION OF LINEAR EQUATIONS SYSTEMS.

3.1. Description of method.-

The equations system

$$\begin{aligned} a_{11}.x_1 + a_{21}.x_2 + \ldots\ldots\ldots + a_{n1}.x_n + \ell_1 &= 0 \\ a_{12}.x_1 + a_{22}.x_2 + \ldots\ldots\ldots + a_{n2}.x_n + \ell_2 &= 0 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & \qquad (8-1) \\ a_{1n}.x_1 + a_{2n}.x_2 + \ldots\ldots\ldots + a_{nn}.x_n + \ell_n &= 0 \end{aligned}$$

with n equations and n unknowns may be represented by means of the cracovian expression

$$\underline{x} \cdot \gamma\underline{a} + \underline{\ell} = 0 \qquad (8-2)$$

in which

$$\underline{x} = \begin{Bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{Bmatrix} ; \quad \underline{a} = \begin{Bmatrix} a_{11} & a_{21} & \dots & a_{n1} \\ a_{12} & a_{22} & \dots & a_{n2} \\ \dots & \dots & \dots & \dots \\ a_{1n} & a_{2n} & \dots & a_{nn} \end{Bmatrix} ; \quad \underline{\ell} = \begin{Bmatrix} \ell_1 \\ \ell_2 \\ \dots \\ \ell_n \end{Bmatrix} \qquad (9 - 1)$$

Cracovian $\underline{x}$ is obtained from

$$\underline{x} = - \underline{\ell} : \underline{a} = - \underline{\ell} \cdot \underline{a}^{-1} \qquad (9 - 2)$$

that is, "the cracovian of the unknowns is equal to the product of the cracovian of constant terms by the inverse of the cracovian of coefficients".- The obtainment of the inverse of the craco - vian of coefficients, like the obtainment of the inverse of a matrix, is a very difficult operation, specially in case of a high number of unknowns. Because of this, it is advisable to use the method of decomposition into two canonical cracovians as explained in 2.10 , page 6 .-

Introducing the cracovians

$$\underline{x}' = \begin{Bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \\ 1 \end{Bmatrix} \qquad \underline{a}' = \begin{Bmatrix} a_{11} & a_{21} & \dots & a_{n1} & \ell_1 \\ a_{12} & a_{22} & \dots & a_{n2} & \ell_2 \\ \dots & \dots & \dots & \dots & \dots \\ a_{1n} & a_{2n} & \dots & a_{nn} & \ell_n \end{Bmatrix} \qquad (9 - 3)$$

the system (8 - 1) may be written

$$\underline{x}' \cdot \gamma \underline{a}' = 0 \qquad (10\text{-}1)$$

Cracovian $\underline{a}'$ may be decomposed into the cracovians

$$\underline{g} = \begin{Bmatrix} 1 & g_{21} & g_{31} & \cdots & g_{n1} & \ell_1' \\ 0 & 1 & g_{32} & \cdots & g_{n2} & \ell_2' \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 1 & \ell_n' \end{Bmatrix} ; \quad \underline{h} = \begin{Bmatrix} h_{11} & h_{21} & \cdots & h_{n1} \\ 0 & h_{22} & \cdots & h_{n2} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & h_{nn} \end{Bmatrix} \qquad (10\text{-}2)$$

such that

$$\underline{a}' = \underline{g} \cdot \underline{h} \qquad (10\text{-}3)$$

Introducing this expression into (10-1)

$$\underline{x}' \cdot \gamma(\underline{g} \cdot \underline{h}) = 0 \qquad (10\text{-}4)$$

Using the properties of cracovian multiplication this equation becomes

$$\begin{aligned} \underline{x}' \cdot (\underline{h} \cdot \underline{g}) &= 0 \\ (\underline{x}' \cdot \gamma \underline{g}) \cdot \underline{h} &= 0 \end{aligned} \qquad (10\text{-}5)$$

As cracovian $\underline{h}$ is different from zero, then

$$\underline{x}' \cdot \gamma_{\underline{g}} = 0 \qquad (11\text{-}1)$$

The cracovian of unknowns is easily obtained fron this craco - vian equation by using the definition of cracovian multiplication. In order to make clear this procedure, it is advisable to develop equation (11-1) supposing $n = 3$

$$\begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{Bmatrix} \cdot \begin{Bmatrix} 1 & 0 & 0 \\ g_{21} & 1 & 0 \\ g_{31} & g_{32} & 1 \\ \ell_1' & \ell_2' & \ell_3' \end{Bmatrix} = 0 \qquad (11\text{-}2)$$

that is

$$\begin{aligned} x_1 + g_{21} \cdot x_2 + g_{31} \cdot x_3 + \ell_1' &= 0 \\ x_2 + g_{32} \cdot x_3 + \ell_2' &= 0 \\ x_3 + \ell_3' &= 0 \end{aligned} \qquad (11\text{-}3)$$

Obviously, the unknowns are obtained from back solution of this system.-

By making the substitutions

$$\ell_i = a_{n+1,i} \quad \text{and} \quad \ell_i' = g_{n+1,i} \qquad (11\text{-}4)$$

and supposing again $n = 3$, the following equality is obtained

$$\begin{Bmatrix} a_{11} & a_{21} & a_{31} & a_{41} \\ a_{12} & a_{22} & a_{32} & a_{42} \\ a_{13} & a_{23} & a_{33} & a_{43} \end{Bmatrix} = \begin{Bmatrix} 1 & g_{21} & g_{31} & g_{41} \\ 0 & 1 & g_{32} & g_{42} \\ 0 & 0 & 1 & g_{43} \end{Bmatrix} \cdot \begin{Bmatrix} h_{11} & h_{21} & h_{31} \\ 0 & h_{22} & h_{32} \\ 0 & 0 & h_{33} \end{Bmatrix} \qquad (12\text{-}1)$$

The elements of cracovians $\underline{g}$ and $\underline{h}$ are obtained in the following way:

a) The first row of $\underline{h}$ is obtained by multiplying the first column of $\underline{g}$ by each column of $\underline{h}$, so

$$h_{11} = a_{11} \; ; \quad h_{21} = a_{12} \; ; \; h_{31} = a_{13} \qquad (12\text{-}2)$$

It can be noticed that the first row of $\underline{h}$ is equal to the first column of $\underline{a}$.-

b) Once the first row of $\underline{h}$ is obtained, the first row of $\underline{g}$ can be obtained by multiplying the first column of $\underline{h}$ by each column of $\underline{g}$, so

$$g_{21} = a_{21}/h_{11} \; ; \quad g_{31} = a_{31}/h_{11} \; ; \; g_{41} = a_{41}/h_{11} \qquad (12\text{-}3)$$

As $h_{11} = a_{11}$, the first row of $\underline{g}$ is equal to the first row of $\underline{a}$ divided by a_{11} .-

c) Using now the second column of $\underline{g}$, the expressions

$$g_{21} \cdot h_{21} + h_{\underline{22}} = a_{22}$$
$$g_{21} \cdot h_{31} + h_{\underline{32}} = a_{23} \qquad (13\text{-}1)$$

make possible to obtain the second row of $\underline{h}$.-

d) The second row of $\underline{g}$ is obtained by multiplying the second column of $\underline{h}$ by each column of $\underline{g}$, that is

$$g_{31} \cdot h_{21} + g_{\underline{32}} \cdot h_{22} = a_{32}$$
$$g_{41} \cdot h_{21} + g_{\underline{42}} \cdot h_{22} = a_{42} \qquad (13\text{-}2)$$

e) The last element of $\underline{h}$ is obtained from

$$g_{31} \cdot h_{31} + g_{32} \cdot h_{32} + h_{\underline{33}} = a_{33} \qquad (13\text{-}3)$$

and the last element of $\underline{g}$ from

$$g_{41} \cdot h_{31} + g_{42} \cdot h_{32} + g_{\underline{43}} \cdot h_{33} = a_{43} \qquad (13\text{-}4)$$

It is remarkable that each element of cracovians $\underline{g}$ and $\underline{h}$ is always obtained from solving an equation with an unknown. Moreover, as all operations are additions of products, they can be accomplished with a calculating machine making no notes of intermediate results.-

In brief, the typical cracovian operation of decomposition into two canonical cracovians, makes possible to obtain the solution

of practically any system of linear equations in a very simple way.-

3.2. THE PROGRAM.

3.2.1. General organization.-

According to the explanations of preceding section, it is very easy to develop general expressions to obtain the elements of cracovians $\underline{g}$ and $\underline{h}$. By considering substitutions (11-4) the first row of $\underline{h}$ is obtained from

$$h_{r1} = a_{1r} \qquad r = 1, 2, \ldots\ldots, n \qquad (14\text{-}1)$$

and the first row of $\underline{g}$ from

$$g_{r1} = a_{r1}/h_{11} \qquad r = 2, 3, \ldots, n, n+1 \qquad (14\text{-}2)$$

The second and other rows of $\underline{h}$ are obtained from

$$h_{sr} = a_{rs} - (g_{r1}.h_{s1} + g_{r2}.h_{s2} + \ldots + g_{r,r-1}.h_{s,r-1})$$

starting from $r = 2$ and $s = r, r+1, \ldots, n$ (14-3)

and the second and other rows of $\underline{g}$ from

$$g_{rs} = \left[a_{rs} - (g_{r1}.h_{s1} + g_{r2}.h_{s2} + \ldots + g_{r,s-1}.h_{s,s-1})\right]/h_{ss}$$

starting from $s = 2$ and $r = s+1, s+2, \ldots, n+1$ (14-4)

These expressions allow the obtainment of the elements of $\underline{g}$ and $\underline{h}$ in alternated way : first and element of $\underline{h}$, later an element of $\underline{g}$ and so on.-

In order to save core storage, the program has been developed in such a way that the elements of $\underline{g}$ and $\underline{h}$ are obtained by - accomplishing a convenient trasformation on cracovian $\underline{a}$. Once this trasformation has been accomplished, cracovian $\underline{a}$ contains:

a) over the principal diagonal, the elements different from zero of cracovian $\underline{g}$, except the diagonal elements whose value is always 1 .

b) in and under the principal diagonal, the elements different from zero of the transpose of $\underline{h}$.-

That is, supposing $n = 3$, cracovian $\underline{a}$ is transformed into

$$\begin{Bmatrix} a_{11} & a_{21} & a_{31} & a_{41} \\ a_{12} & a_{22} & a_{32} & a_{42} \\ a_{13} & a_{23} & a_{33} & a_{43} \end{Bmatrix} \rightarrow \begin{Bmatrix} h_{11} & g_{21} & g_{31} & g_{41} \\ h_{21} & h_{22} & g_{32} & g_{42} \\ h_{31} & h_{32} & h_{33} & g_{43} \end{Bmatrix} \qquad (15\text{-}1)$$

In order to achieve this transformation, expressions (14-2) , (14-3) and (14-4) have been changed into

$$a_{r1} \leftarrow a_{r1}/a_{11} \qquad r = 2, 3, \ldots, n+1 \qquad (15\text{-}2)$$

$$a_{rs} \leftarrow a_{rs} - (a_{r1} \cdot a_{1s} + a_{r2} \cdot a_{2s} + \ldots + a_{r,r-1} \cdot a_{r-1,s}) \qquad (15\text{-}3)$$

starting from $r = 2$ and $s = r, r+1, \ldots, n$

$$a_{rs} \leftarrow \left[a_{rs} - (a_{r1} \cdot a_{1s} + a_{r2} \cdot a_{2s} + \ldots + a_{r,s-1} \cdot a_{s-1,s}) \right] / a_{ss}$$

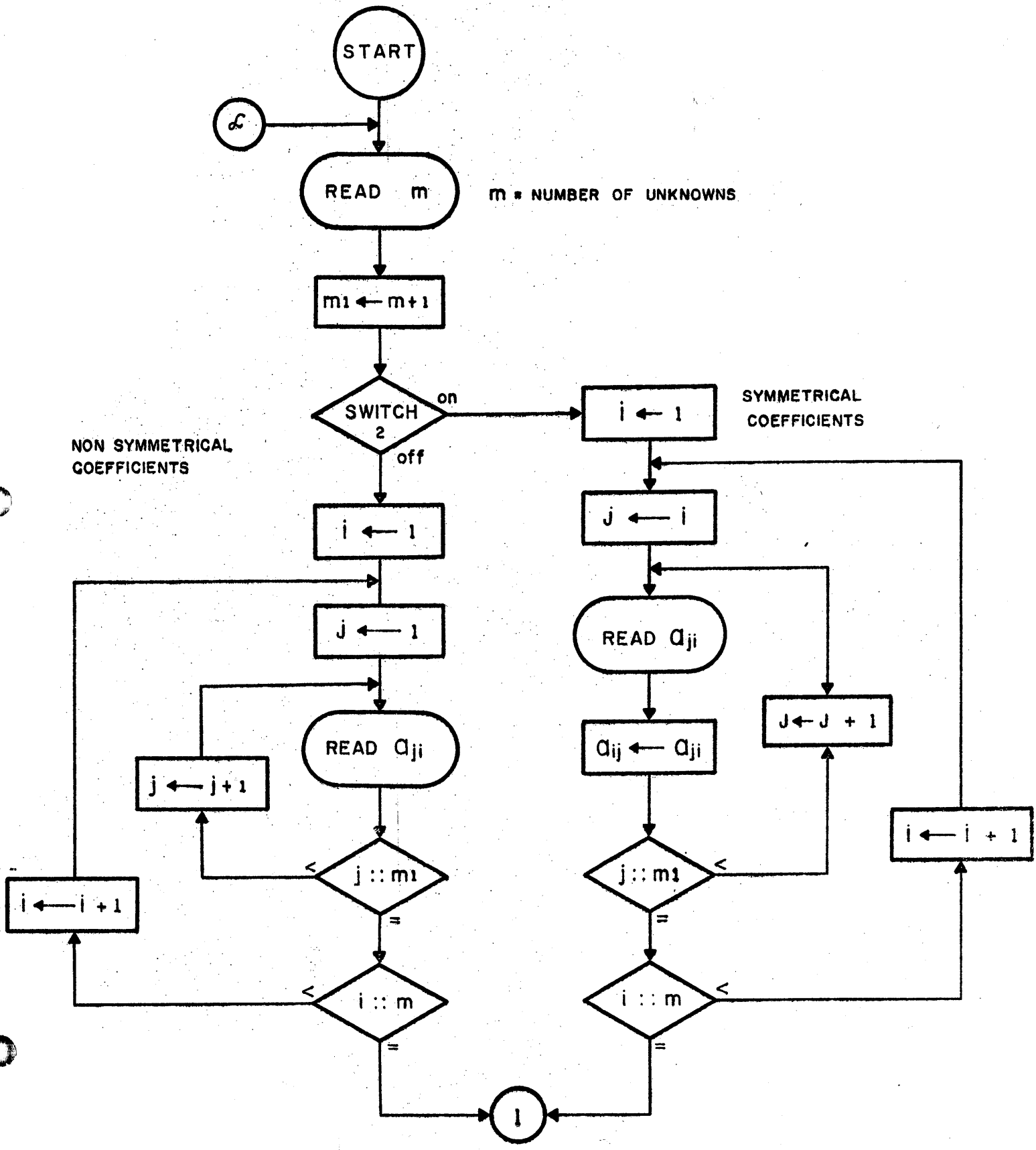
starting from $s = 2$ and $r = s+1, s+2, \ldots, n+1$ (16-1)

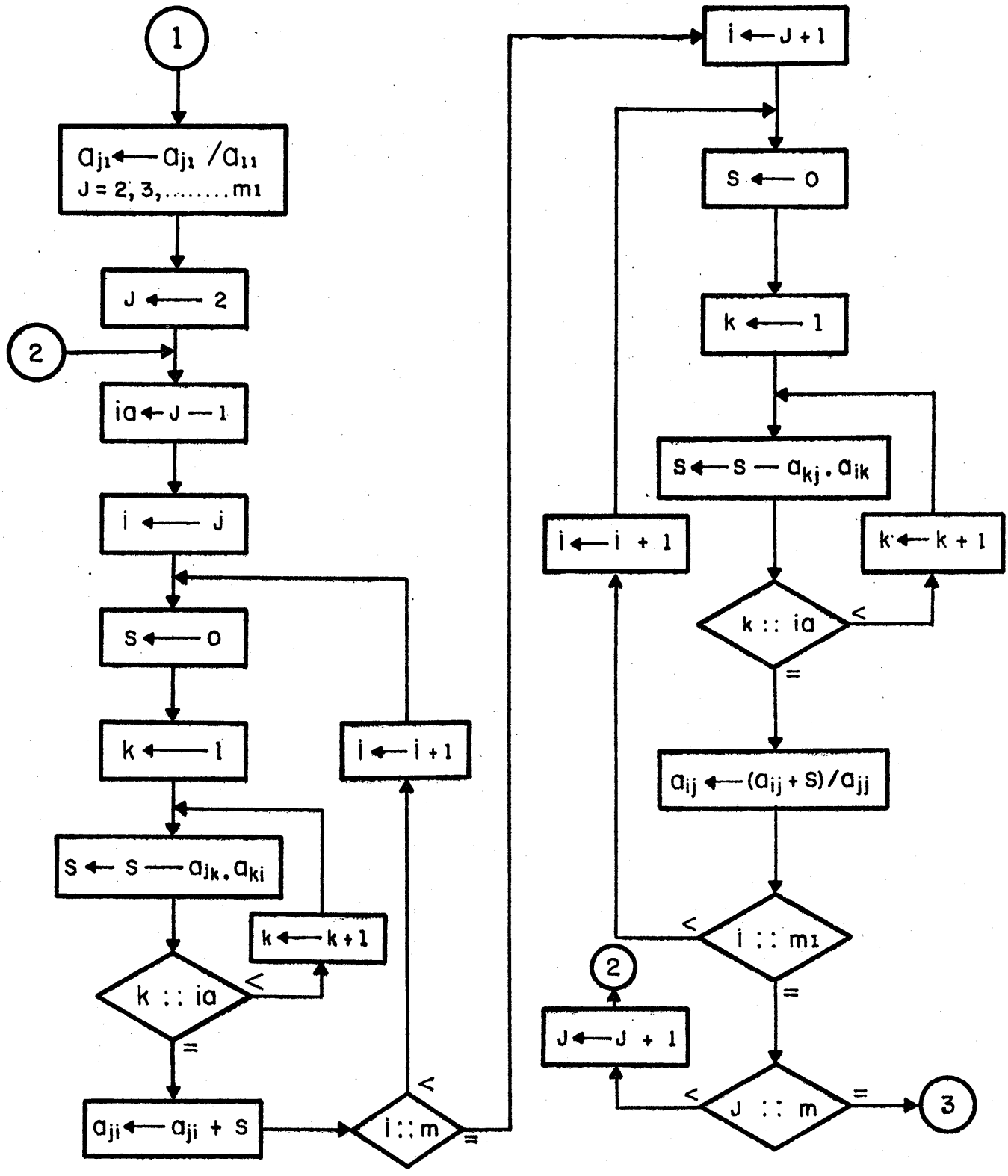
By means of this procedure it is only necessary to keep cracovian $\underline{a}$. As explained in page 11, obtainment of unknowns is inmediate once the cracovian $\underline{g}$ has been obtained.-

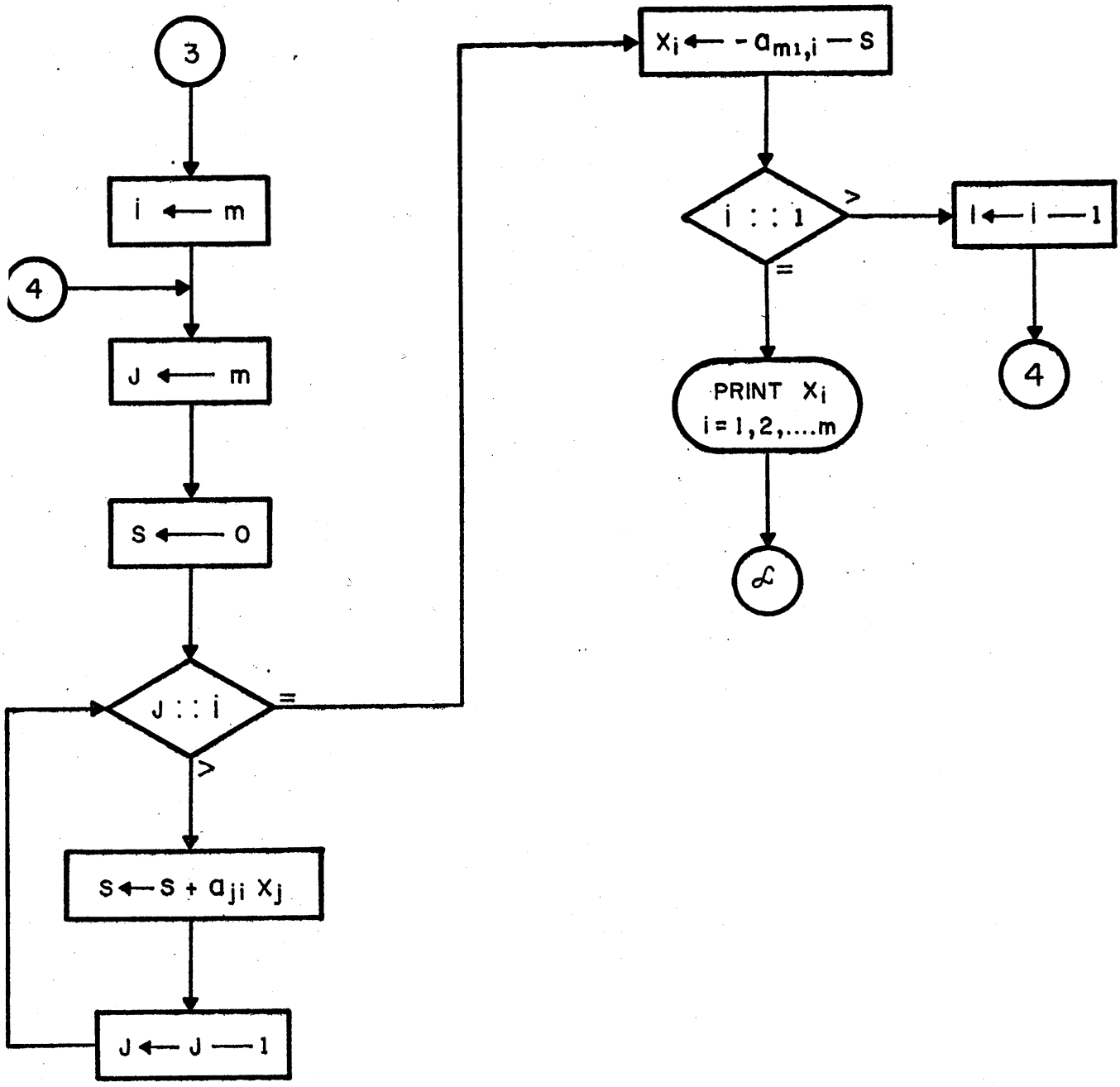
The flow chart and program listing may help the reader to make clear the whole procedure of decomposition and to study all the possibilities of the program.-

3.2.2 FLOW CHART OF PROGRAM I

(General Solution of Linear Equations Systems)




1
$a_{j1} \leftarrow a_{j1} / a_{11}$
J = 2, 3, ... m1
J ← 2
2
ia ← J − 1
i ← j
s ← 0
k ← 1
s ← s − $a_{jk} . a_{ki}$
k ← k + 1
k :: ia
<
=
$a_{ji} \leftarrow a_{ji} + s$
i ← i + 1
i :: m
<
=
i ← J + 1
s ← 0
k ← 1
s ← s − $a_{kj} . a_{ik}$
i ← i + 1
k ← k + 1
k :: ia
<
=
$a_{ij} \leftarrow (a_{ij} + S) / a_{jj}$
i :: m1
<
=
2
J ← J + 1
J :: m
<
=
3


3
i ← m
4
J ← m
S ← 0
J : : i
=
>
S ← S + a_ji X_j
J ← J — 1
X_i ← - a_m1,i — S
i : : 1
>
=
I ← i — 1
4
PRINT X_i
i = 1, 2, ... m
α

3.2.3. Program listing and solution of a numerical example

```
C
C      DECOMPOSITION OF A CRACOVIAN INTO TWO CANONICAL CRACOVIANS
C      AND SOLUTION OF GENERAL LINEAR EQUATIONS SYSTEMS
C
       DIMENSION A(27,26), X(26)
     1 READ 100,M
       PRINT 101,M
       M1=M+1
       IF(SENSE SWITCH 2)2,4
     2 DO 3 I=1,M
       DO 3 J=I,M1
       READ 102, A(J,I)
     3 A(I,J)=A(J,I)
       GO TO 6
     4 DO 5 I=1,M
       DO 5 J=1,M1
     5 READ 102,A(J,I)
     6 IF(SENSE SWITCH 1)7,15
     7 PRINT 103
       IF(M-6)8,8,12
     8 DO 9 I=1,M
     9 TYPE 104,I
       TYPE 105
       DO 11 I=1,M
       DO 10 J=1,M1
    10 TYPE 102,A(J,I)
    11 CONTROL 102
       GO TO 15
    12 DO 14 I=1,M
       DO 13 J=1,M1
    13 PRINT 102,A(J,I)
    14 CONTROL 102
C
C      DECOMPOSITION OF CRACOVIAN A
C
    15 DO 16 J=2,M1
    16 A(J,1)=A(J,1)/A(1,1)
       DO 20 J=2,M
       IA=J-1
       DO 18 I=J,M
       S=0.
       DO 17 K=1,IA
    17 S=S-A(J,K)*A(K,I)
    18 A(J,I)=A(J,I)+S
       J1=J+1
       DO 20 I=J1,M1
       S=0.
       DO 19 K=1,IA
    19 S=S-A(K,J)*A(I,K)
    20 A(I,J)=(A(I,J)+S)/A(J,J)
```

```
C
C     CALCULATION OF UNKNOWNS
C
      I=M
   21 J=M
      S=0.
   22 IF(J-I)23,24,23
   23 S=S+A(J,I)*X(J)
      J=J-1
      GO TO 22
   24 X(I)=(-A(M1,I)-S)
      IF(I-1)25,26,25
   25 I=I-1
      GO TO 21
   26 IF(SENSE SWITCH 1)27,44
   27 PRINT 106
      IF(M-6)28,28,36
   28 ONE=1.
      DO 31 I=1,M
      TYPE 102,ONE
      I1=I+1
      DO 29 J=I1,M1
   29 TYPE 102, A(J,I)
      CONTROL 102
      IF(I-M)30,32,30
   30 DO 31 K=1,I
   31 TYPE 107
   32 PRINT 108
      DO 35 I=1,M
      DO 33 J=I,M
   33 TYPE 102,A(I,J)
      IF(I-M)34,44,34
   34 CONTROL 102
      DO 35 K=1,I
   35 TYPE 107
   36 ZERO=0.
      DO 39 I=1,M
      PRINT 102,ONE
      I1=I+1
      DO 37 J=I1,M1
   37 PRINT 102,A(J,I)
      CONTROL 102
      IF(I-M)38,40,38
   38 DO 39 K=1,I
   39 PRINT 102,ZERO
```

```
 40 PRINT 108
    DO 43 I=1,M
    DO 41 J=I,M
 41 PRINT 102,A(I,J)
    IF(I-M)42,44,42
 42 CONTROL 102
    DO 43 K=1,I
 43 PRINT 102,ZERO
 44 PRINT 109
    DO 45 I=1,M
 45 PRINT 110,I,X(I)
    PAUSE
    GO TO 1
100 FORMAT(I4)
101 FORMAT(//10HUNKNOWNS = I3)
102 FORMAT(E11.4)
103 FORMAT(//25HCOEFFICIENTS OF EQUATIONS//)
104 FORMAT(4X2HX(,I2,3H ) )
105 FORMAT(3X7HC. TERM//)
106 FORMAT(//11HCRACOVIAN G//)
107 FORMAT(11X)
108 FORMAT(//11HCRACOVIAN H//)
109 FORMAT(//10X8HUNKNOWNS//)
110 FORMAT(2HX(,I3,2H)=E11.4)
    END
```

UNKNOWNS = 6

COEFFICIENTS OF EQUATIONS

X(1)	X(2)	X(3)	X(4)	X(5)	X(6)	C. TERM
.6518E 03	-.2392E 03	.9460E 02	.1880E 03	.1480E 03	.5800E 02	-.4315E 03
-.1196E 03	.8867E 04	-.9610E 03	-.2260E 03	-.5150E 03	.1860E 03	-.1885E 03
.9460E 02	-.1922E 04	.2439E 05	.4740E 03	-.4820E 04	.5920E 03	-.8270E 02
-.9390E 02	-.2260E 03	.2370E 03	.4810E 05	-.2370E 04	-.2015E 04	-.1200E 03
.7420E 02	-.5150E 03	-.2410E 04	-.2370E 04	.7860E 05	-.6420E 04	-.5250E 02
.5800E 02	.3720E 03	.5920E 03	-.4030E 04	-.1284E 05	.1585E 06	-.3350E 02

CRACOVIAN G

.1000E 01	-.3669E 00	.1451E 00	.2884E 00	.2270E 00	.8898E-01	-.6620E 00
	.1000E 01	-.1069E 00	-.2170E-01	-.5529E-01	.2228E-01	-.3033E-01
		.1000E 01	.1678E-01	-.2045E 00	.2588E-01	-.3198E-02
			.1000E 01	-.4816E-01	-.4170E-01	-.3935E-02
				.1000E 01	-.8276E-01	-.4539E-03
					.1000E 01	-.1880E-04

CRACOVIAN H

.6518E 03	-.1196E 03	.9460E 02	-.9390E 02	.7420E 02	.5800E 02
	.8823E 04	-.1887E 04	-.2604E 03	-.4877E 03	.3932E 03
		.2417E 05	.2227E 03	-.2472E 04	.6256E 03
			.4811E 05	-.2360E 04	-.4048E 04
				.7793E 05	-.1289E 05
					.1572E 06

UNKNOWNS

X(1)= .6715E 00
X(2)= .3079E-01
X(3)= .3225E-02
X(4)= .3957E-02
X(5)= .4554E-03
X(6)= .1880E-04

4. SOLUTION OF NORMAL EQUATIONS SYSTEMS.-

4.1. Description of method.

4.1.1. Obtainment of unknowns.

Application of least sguares adjustment leads to linear equations systems of symmetrical coefficients called "normal equations systems". Although in this case it is also possible to use the method of decomposition into two canonical cracovians explained in section 3, it is advisable to use a special method which provides the possibility to obtain the mean square errors of unknowns in a very simple way. This special method consists in decomposition of cracovian of coefficients into two equal canonical cracovians and it is called "the cracovian root method".-

The system of error equations

$$v_i = a_i . x_1 + b_i . x_2 + ... + g_i . x_m + \ell_i \qquad (24\text{-}1)$$
$$i = 1, 2, ... , n$$

may be written very conveniently in cracovian notation as

$$\underline{v} = \underline{x} . \gamma\underline{a} + \underline{\ell} \qquad (24\text{-}2)$$

in Which

$$\underline{v} = \begin{Bmatrix} v_1 \\ v_2 \\ .. \\ v_n \end{Bmatrix}; \underline{a} = \begin{Bmatrix} a_1 & b_1 & \dots & g_1 \\ a_2 & b_2 & \dots & g_2 \\ \dots & \dots & \dots & .. \\ a_n & b_n & \dots & g_n \end{Bmatrix}; \underline{x} = \begin{Bmatrix} x_1 \\ x_2 \\ .. \\ x_n \end{Bmatrix}, \underline{\ell} = \begin{Bmatrix} \ell_1 \\ \ell_2 \\ .. \\ \ell_n \end{Bmatrix} \qquad (25\text{-}1)$$

Introducing the diagonal cracovian

$$\underline{p} = \begin{Bmatrix} p_1 & 0 & \dots & 0 \\ 0 & p_2 & \dots & 0 \\ \dots & \dots & \dots & \\ 0 & 0 & \dots & p_n \end{Bmatrix} \qquad (25\text{-}2)$$

in which the p_i are the weight coefficients of observations, it is obvious that cracovian expression

$$\underline{x} \,.[\,(\underline{a} \,.\, \underline{p}) \,.\, \underline{a}\,] + \underline{\ell} \,.\, (\underline{a} \,.\, \underline{p}) = 0 \qquad (25\text{-}3)$$

represents the normal equations system derived from system (24-1).- Making the substitutions

$$\underline{A} = (\underline{a} \,.\, \underline{p}) \,.\, \underline{a} \qquad \underline{L} = \underline{\ell} \,.\, (\underline{a} \,.\, \underline{p}) \qquad (25\text{-}4)$$

expression (25 -3) becomes

$$\underline{x} \cdot \underline{A} + \underline{L} = 0 \qquad (26\text{-}1)$$

Developing this last equation, the following system is obtained

$$\begin{aligned}
&A_{11} \cdot x_1 + A_{21} \cdot x_2 + \ldots + A_{m,1} \cdot x_m + L_1 = 0 \\
&A_{21} \cdot x_1 + A_{22} \cdot x_2 + \ldots + A_{m,2} \cdot x_m + L_2 = 0 \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \qquad (26\text{-}2) \\
&A_{m,1}\, x_1 + A_{m,2} \cdot x_2 + \ldots + A_{m,m} \cdot x_m + L_m = 0
\end{aligned}$$

where, in agreement with classical Gauss notation

$$A_{11} = [paa] \; ; \; A_{21} = [pab] \; ; \; \ldots \; ; \; A_{m,1} = [pag] ; \; L_1 = [pal]$$
$$A_{22} = [pbb] \quad \text{and so on.-}$$

Let $m' = m + 1$ and $L_i = A_{m',i}$; then the normal equations system (26 -2) may be enlarged and transformed into

$$\begin{aligned}
&A_{11} \cdot x_1 + A_{21} \cdot x_2 + \ldots + A_{m,1} \cdot x_m + A_{m',1} = 0 \\
&A_{21} \cdot x_1 + A_{22} \cdot x_2 + \ldots + A_{m,2} \cdot x_m + A_{m',2} = 0 \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \qquad (26\text{-}3) \\
&A_{m,1} \cdot x_1 + A_{m,2} \cdot x_2 + \ldots + A_{m,m} \cdot x_m + A_{m',m} = 0 \\
&A_{m',1} \cdot x_1 + A_{m',2} \cdot x_2 + \ldots + A_{m',m} \cdot x_m + A_{m',m'} = [pvv]
\end{aligned}$$

The last added equation, equivalent to the known relation

$$[pal]\ x_1 + [pbl]\ x_2 + \ \ldots\ + [pgl]\ x_m + [pll] = [pvv]$$

will provide the very important control of [pvv] as explained later.- Introducing the cracovians

$$x' = \left\{ \begin{matrix} x_1 \\ x_2 \\ \ldots \\ x_m \\ 1 \end{matrix} \right\} ; \quad A' = \left\{ \begin{matrix} A_{11} & A_{21} & \ldots & A_{m,1} & A_{m',1} \\ A_{21} & A_{22} & \ldots & A_{m,2} & A_{m',2} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ A_{m,1} & A_{m,2} & \ldots & A_{m,m} & A_{m',m} \\ A_{m',1} & A_{m',2} & \ldots & A_{m',m} & A_{m',m'} \end{matrix} \right\} ; \quad C = \left\{ \begin{matrix} 0 \\ 0 \\ \ldots \\ 0 \\ [pvv] \end{matrix} \right\}$$

system (26 - 3) may be written as

$$\underline{x}' \ . \ \underline{A}' \ = \ \underline{C} \qquad (27\text{-}1)$$

Cracovian A' may be decomposed into two equal canonical cracovians such that

$$\underline{A}' \ = \ \underline{B} \ . \ \underline{B} \ = \ \underline{B}^2 \qquad (27\text{-}2)$$

As cracovian $\underline{A}'$ is symmetric, according to 2.11, page 6

$$\underline{B} = \begin{Bmatrix} B_{11} & B_{21} & \dots & B_{m,1} & B_{m',1} \\ 0 & B_{22} & \dots & B_{m,2} & B_{m',2} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & B_{m,m} & B_{m',m} \\ 0 & 0 & \dots & 0 & B_{m',m'} \end{Bmatrix}$$

The elements of cracovian B are obtained by applying the definition of cracovian multiplication by means of the following relations

$$B_{11} = \sqrt{A_{11}}; \quad B_{r1} = A_{r1}/A_{11} \quad r = 2,3, \dots , m' \qquad (28\text{-}1)$$

$$B_{rr} = \sqrt{A_{rr} - (B_{r1}^2 + B_{r2}^2 + \dots + B_{r,r-1}^2)}$$

$$r = 2, 3, \dots , m' \qquad (28\text{-}2)$$

$$B_{rs} = [A_{rs} - (B_{s1} \cdot B_{r1} + B_{s2} \cdot B_{r2} + \dots + B_{s,s-1} \cdot B_{r,s-1})] / B_{ss}$$

starting from $s = 2$ and $r = s + 1, s + 2, \dots , m'$

(28-3)

Introducing (27 -2) into (27-1)

$$\underline{x}' \cdot (\underline{B} \cdot \underline{B}) = \underline{C}$$

or $$(\underline{x}' \cdot \tau\underline{B}) \cdot \underline{B} = \underline{C}$$

Multiplyng both sides by $\gamma_{\underline{B}}^{-1}$

$$[(\underline{x}' \cdot \gamma_{\underline{B}}) \cdot \underline{B}] \cdot \gamma_{\underline{B}}^{-1} = \underline{C} \cdot \gamma_{\underline{B}}^{-1}$$

According to 2.7, page 5, this expression may be written as

$$(\underline{x}' \cdot \gamma_{\underline{B}}) \cdot (\gamma_{\underline{B}}^{-1} \cdot \gamma_{\underline{B}}) = \underline{C} \cdot \gamma_{\underline{B}}^{-1}$$
$$(\underline{x}' \cdot \gamma_{\underline{B}}) \cdot \gamma = \underline{C} \cdot \gamma_{\underline{B}}^{-1}$$

that is

$$\underline{x}' \cdot \gamma_{\underline{B}} = \underline{C} \cdot \gamma_{\underline{B}}^{-1} \qquad (29\text{-}1)$$

If $\underline{C}' = \underline{C} \cdot \gamma_{\underline{B}}^{-1}$, then

$$C' = \begin{Bmatrix} 0 \\ 0 \\ \cdots \\ [pvv] \end{Bmatrix} \cdot \begin{pmatrix} B_{11}^{-1} & y_{12} & \cdots & y_{1,m'} \\ 0 & B_{22}^{-1} & \cdots & y_{2,m'} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & B_{m',m'}^{-1} \end{pmatrix} = \begin{Bmatrix} 0 \\ 0 \\ \cdots \\ [pvv] \cdot B_{m',m'}^{-1} \end{Bmatrix}$$

Developing (29 -1), the following system is obtained

$$
\begin{aligned}
B_{11}\cdot x_1 + B_{21}\cdot x_2 + \dots + B_{m,1}\cdot x_m + B_{m',1} &= 0 \\
B_{22}\cdot x_2 + \dots + B_{m,2}\cdot x_m + B_{m',2} &= 0 \\
\dots\dots\dots\dots\dots\dots\dots\dots\dots & \\
B_{m,m}\cdot x_m + B_{m',m} &= 0 \\
B_{m',m'} = [pvv] \cdot B^{-1}_{m',m'} &
\end{aligned}
$$

Obviously, unknowns are obtained from back solution of this system and the last relation gives the control

$$[pvv] = B^2_{m',m'}$$

It is remarkable that cracovian of normal equations coefficients is equal to the square of cracovian of error equations coefficients which is different from zero. Thus, it is always possible to obtain the cracovian root in case of geodetic problems.-

4.1.2. Obtainment of mean square errors of unknowns.-

Mean square error of unit weight observation may be obtained from

$$m_o = \sqrt{[pvv] / (n - m)} \qquad (30-1)$$

or

$$m_o = B_{m',m'} / \sqrt{n - m} \qquad (30-2)$$

Weight coefficients are then used to obtain mean square errors of unknowns from

$$m_{x_i} = m_o \sqrt{Q_{ii}} \qquad (31\text{-}1)$$

If F is a linear function of unknowns such that

$$F = f_1 . x_1 + f_2 . x_2 + \ldots + f_m . x_m \qquad (31\text{-}2)$$

it may be written in cracovian notation as

$$\underline{F} = \underline{x} . \underline{f} \qquad (31\text{-}3)$$

The normal equations system in case of uniform weight observations may be written as

$$\underline{x} . (\underline{a}^2) + \underline{\ell} . \underline{a} = 0 \qquad (31\text{-}4)$$

Multiplying both sides by $(\underline{a}^2)^{-1}$

$$[\underline{x} . (\underline{a}^2)] . (\underline{a}^2)^{-1} + (\underline{\ell} . \underline{a}) . (\underline{a}^2)^{-1} = 0 \qquad (31\text{-}5)$$

$$\underline{x} + \underline{\ell} [(\underline{a}^2)^{-1} . \gamma \underline{a}] = 0 \qquad (31\text{-}6)$$

$$\underline{x} = - \underline{\ell} . [(\underline{a}^2)^{-1} . \gamma \underline{a}] \qquad (31\text{-}7)$$

Introducing this value of $\underline{x}$ in (31 - 3)

$$F = \{ - \underline{\ell} . [(\underline{a}^2)^{-1} . \gamma \underline{a}] \} . \underline{f}$$

$$F = - \underline{\ell} . \{ \underline{f} [\gamma \underline{a} . (\underline{a}^2)^{-1}] \}$$

Application of law of errors propagation leads to

$$m_F^2 = m_o^2 . \{ \underline{f} . [\gamma \underline{a} . (\underline{a}^2)^{-1}] \}^2$$

$$= m_o^2 . \{ \underline{f} . [(\underline{a}^2)^{-1} . \gamma \underline{a}]^2 \underline{f} \}$$

$$= m_o^2 . \{ [\underline{f} . (\underline{a}^2)^{-1}] . \underline{f} \}$$

This expression may be written as

$$m_F = m_o \sqrt{(\underline{f} . \underline{A}^{-1}) . \underline{f}} \qquad (32\text{-}1)$$

To obtain the mean square error of x_i it is only necessary to stablish $F = x_i$, that is

$$f_i = \begin{Bmatrix} 0 \\ 0 \\ \cdots \\ 1 \\ 0 \end{Bmatrix} \text{ in i-th row}$$

Expression (32- 1) becomes

$$m_{x_i} = m_o \sqrt{(\underline{f}_i \circ \underline{A}^{-1}) \cdot \underline{f}_i}$$

$$= m_o \sqrt{D_{ii}}$$

This means that weight co-efficient of i-th unknown is equal to the element of the inverse of cracovian of coefficients which stands in the i-th row and column, that is to say that diagonal elements of the inverse of cracovian of coefficients are the weight coefficients of unknowns.-

Calculation of these elements is made in a very simple way by calculating first the inverse of the cracovian root of A because

$$\underline{A}^{-1} = (\underline{B}^{-1})^2 \qquad (33\text{-}1)$$

and obtainment of the inverse of a canonical cracovian is inmediate.-

Let $\underline{D} = \underline{B}^{-1}$; then, the emements of $\underline{D}$ are obtained from

$$D_{rr} = 1/B_{rr} \qquad r = 1,2, \ldots, m$$

$$D_{rs} = 0 \qquad \text{if } r \; s$$

$$D_{rs} = -(B_{s1} \cdot D_{r1} + B_{s2} \cdot D_{r2} + \ldots + B_{s,s-1} D_{r,s-1})/B_{ss}$$

startin from r =1 and s = r + 1, r + 2, ..., m

4.2. The program

4.2.1 General crganization

In order to save core storage, the program has been developed in such a way that it is only necessary to keep coefficients of normal equations in and over principal diagonal by means of an unidimensional array.-

The correspondence in nomenclature is

$$A_{rs} = CR_t$$

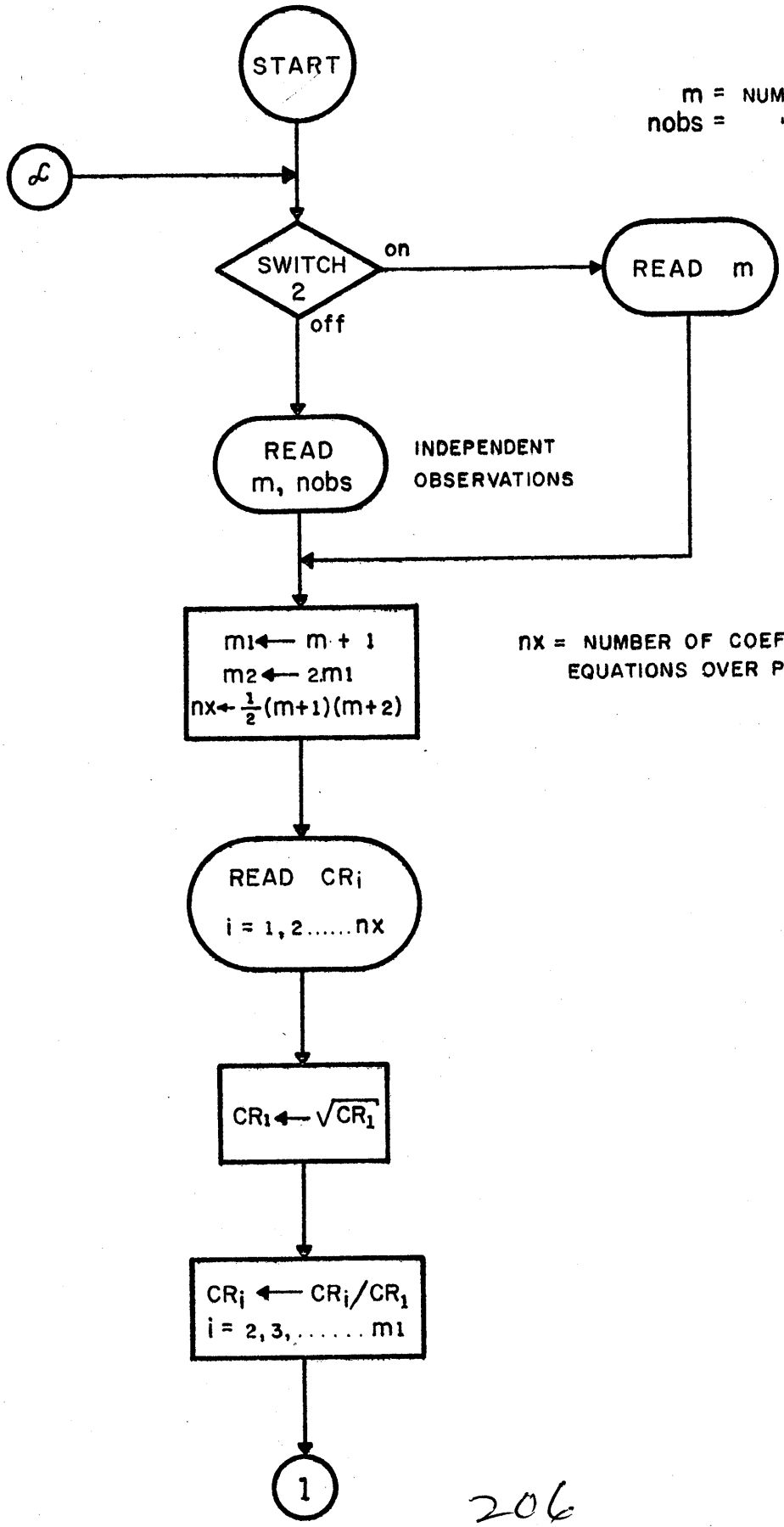
in which $t = r + (s - 1)(m + 1 - s/2)$

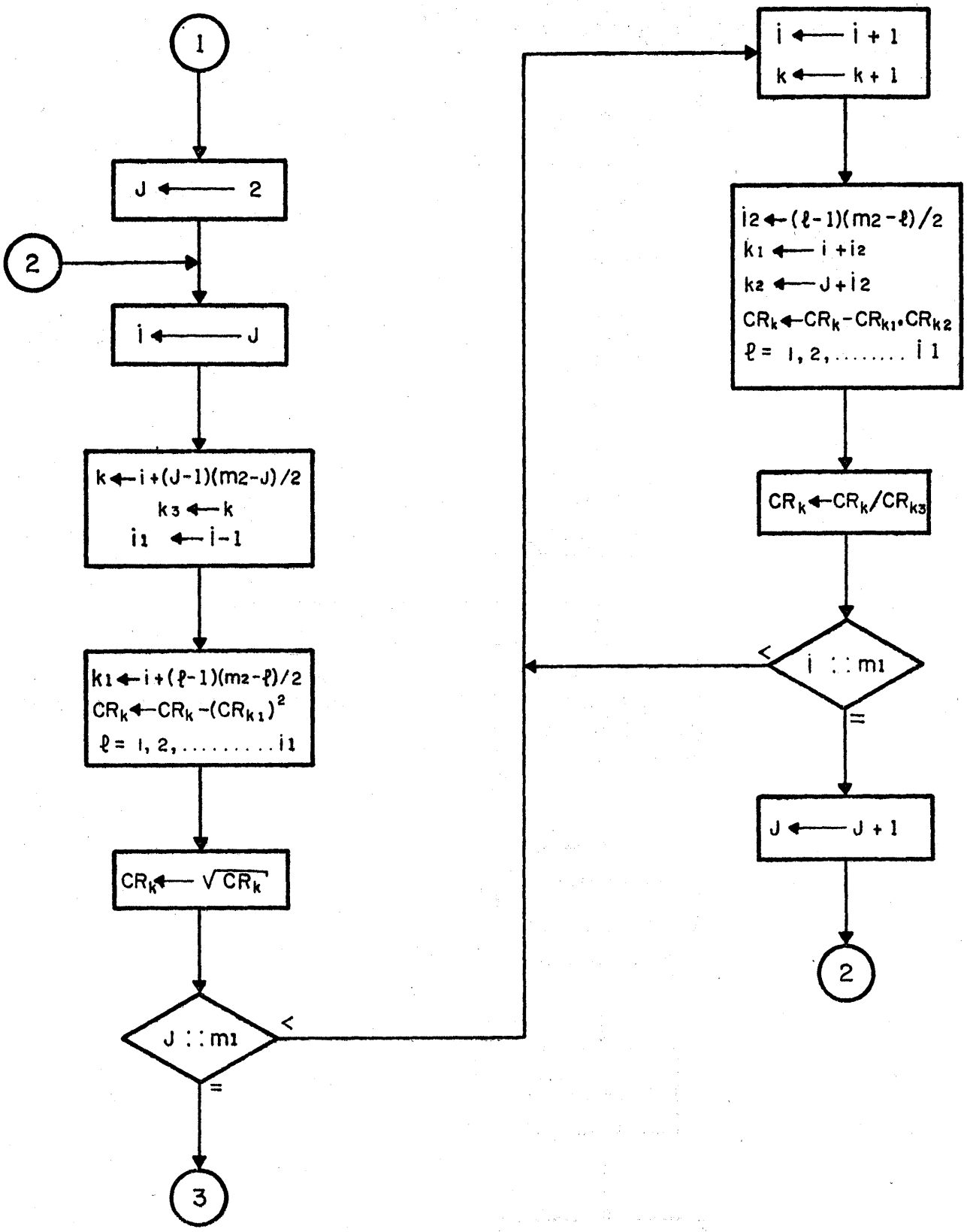
In the first part of program, the CR_i are transformed into the elements of cracovian root and the unknowns are obtained. In the second part, the CR_i are again transformed into the elements of the inverse of cracovian root and the mean square errors of unknowns are calculated.-

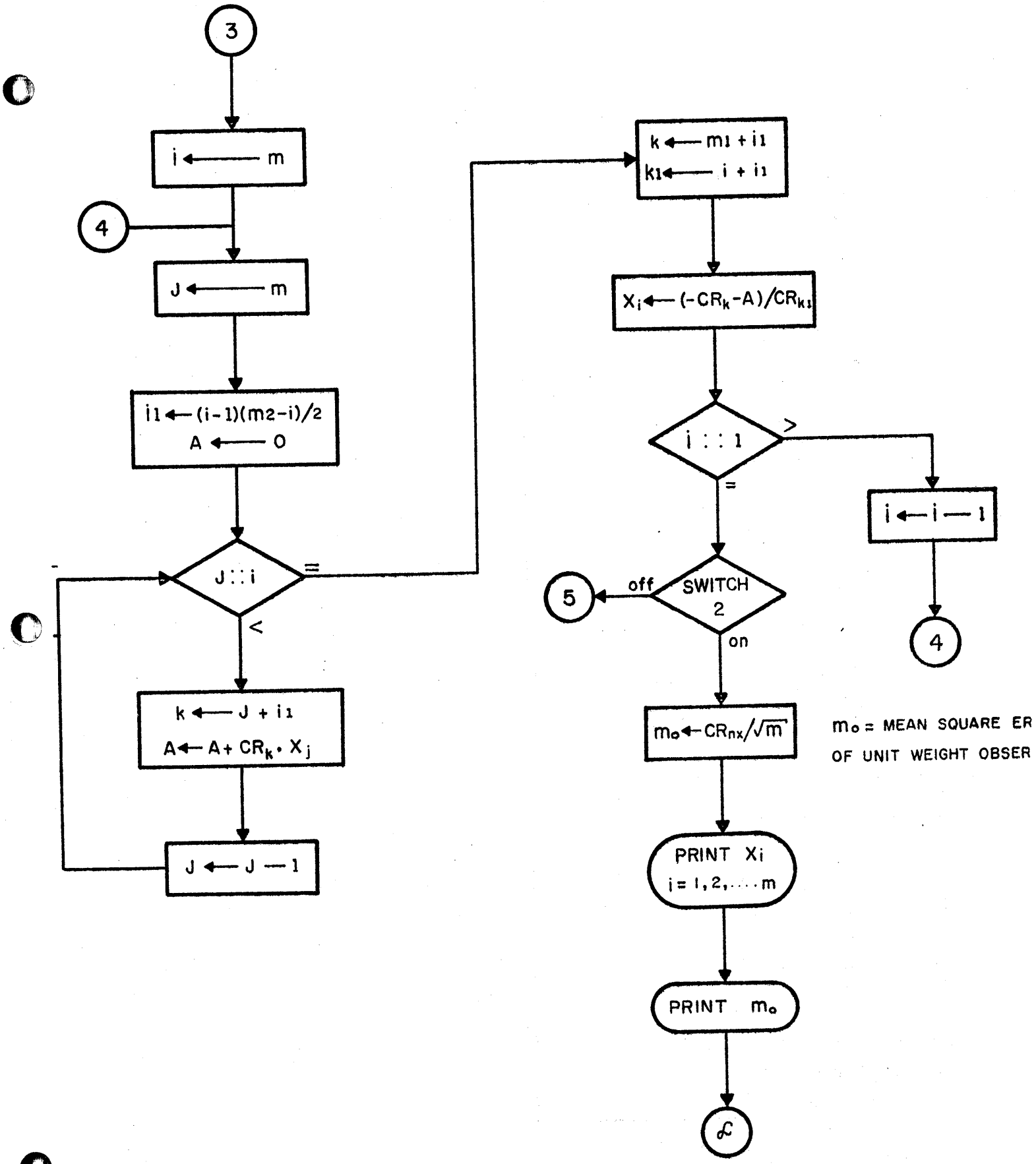
Flow chart and program listing may help the reader to understand better these short explanations.

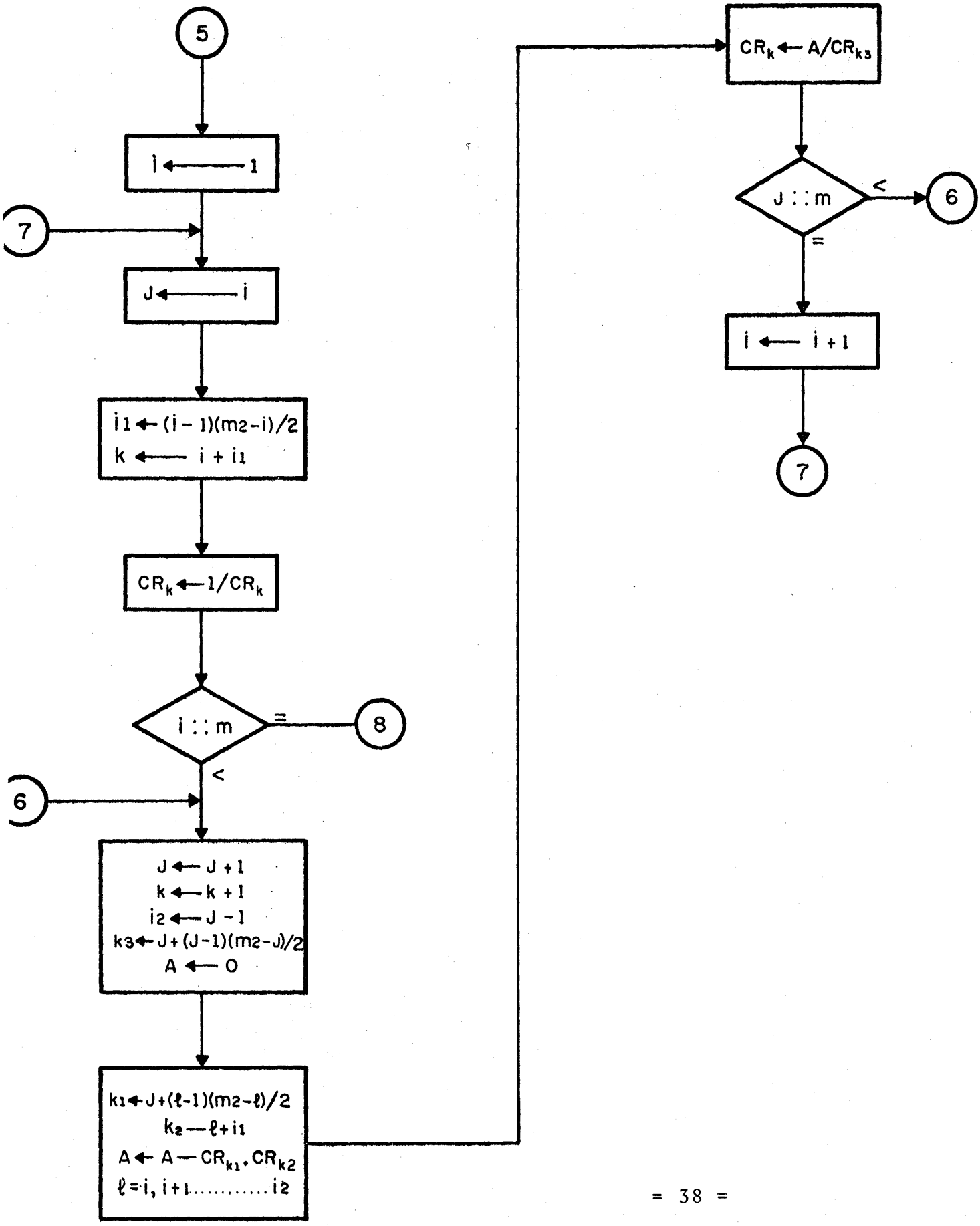
4.2.2 FLOW CHART OF PROGRAM II

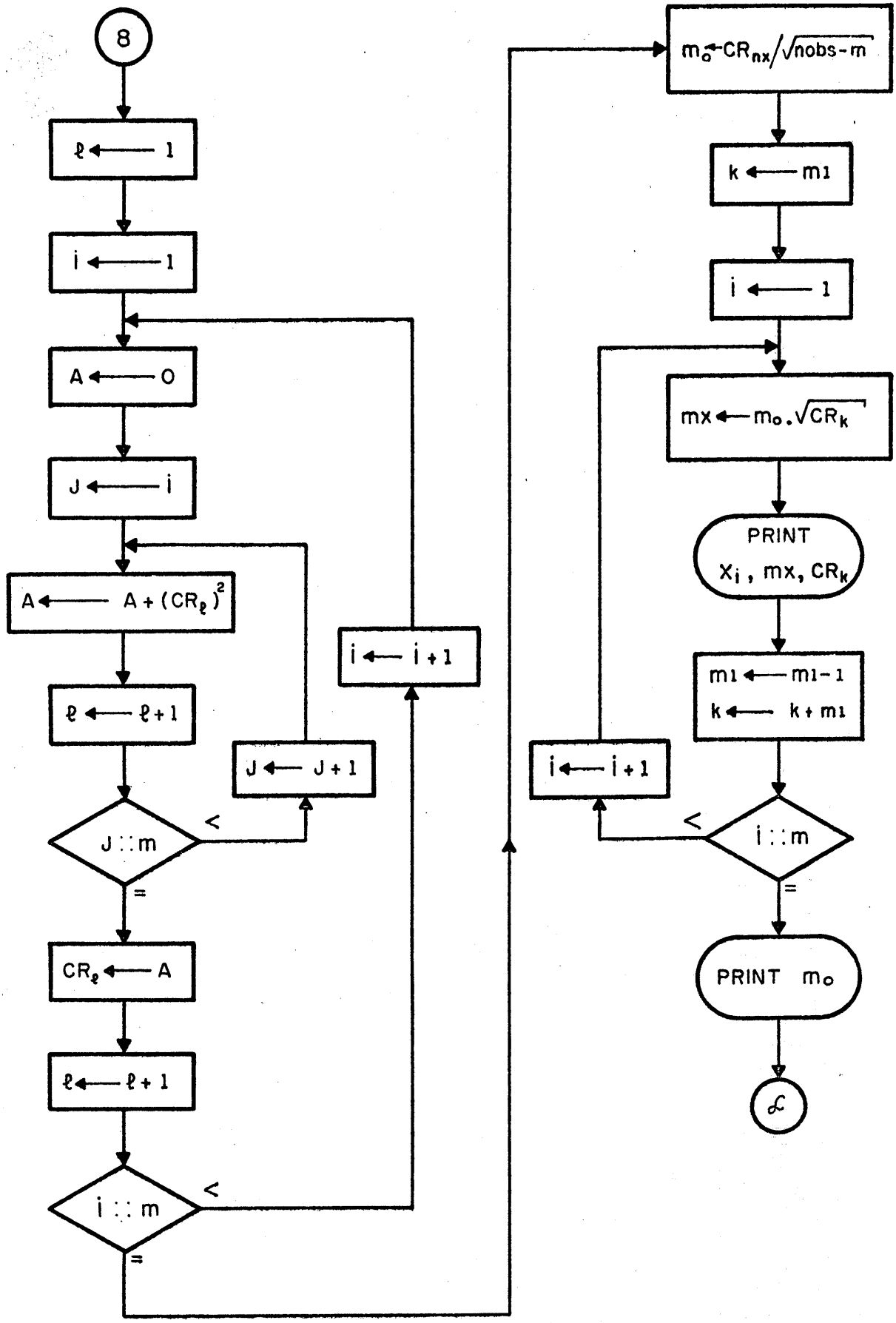
(Solution of Normal Equations Systems)




1
J ← 2
2
i ← J
k ← i+(J-1)(m2-J)/2
k3 ← k
i1 ← i-1
k1 ← i+(ℓ-1)(m2-ℓ)/2
CRk ← CRk-(CRk1)²
ℓ = 1, 2, ... i1
CRk ← √CRk
J :: m1
<
=
3
i ← i + 1
k ← k + 1
i2 ← (ℓ-1)(m2-ℓ)/2
k1 ← i + i2
k2 ← J + i2
CRk ← CRk - CRk1 . CRk2
ℓ = 1, 2, ... i1
CRk ← CRk/CRk3
i :: m1
<
=
J ← J + 1
2


3
i ← m
4
J ← m
i1 ← (i-1)(m2-i)/2
A ← 0
J : : i
=
<
k ← J + i1
A ← A + CRk · Xj
J ← J — 1
k ← m1 + i1
k1 ← i + i1
Xi ← (-CRk-A)/CRk1
i : : 1
>
=
i ← i — 1
4
5
off
SWITCH 2
on
mo ← CRnx/√m
mo = MEAN SQUARE ER
OF UNIT WEIGHT OBSER
PRINT Xi
i = 1, 2, ... m
PRINT mo
£


5
i ← 1
7
J ← i
i1 ← (i − 1)(m2−i)/2
k ← i + i1
CRk ← 1/CRk
i :: m
=
8
<
6
J ← J + 1
k ← k + 1
i2 ← J − 1
k3 ← J + (J−1)(m2−J)/2
A ← 0
k1 ← J + (ℓ−1)(m2−ℓ)/2
k2 ← ℓ + i1
A ← A − CRk1 . CRk2
ℓ = i, i+1 ..... i2
CRk ← A/CRk3
J :: m
<
6
=
i ← i + 1
7


8
ℓ ← 1
i ← 1
A ← 0
J ← i
A ← A + (CRℓ)²
ℓ ← ℓ+1
J ← J+1
J :: m
<
=
CRℓ ← A
ℓ ← ℓ+1
i :: m
<
=
i ← i+1
mo ← CRnx/√nobs-m
k ← m1
i ← 1
mx ← mo.√CRk
PRINT
Xi , mx, CRk
m1 ← m1-1
k ← k + m1
i ← i+1
i :: m
<
=
PRINT mo
ℒ

4.2.3. Program listing and solution of a numerical example

```
C       CRACOVIAN ALGORITHM FOR SOLVING NORMAL EQUATIONS
C       6-11-65  *  PDQ FORTRAN
C
        DIMENSION CR(496), X(30)
      1 IF(SENSE SWITCH 2)2,3
      2 READ 100,M
        PRINT 112
        TYPE 100,M
        GO TO 4
      3 READ 100,M,NOBS
        PRINT 103,M,NOBS
      4 M1=M+1
        M2=2*M1
        NX=(M+2)*(M+1)/2
        DO 5 I=1,NX
      5 READ 101,CR(I)
        IF(SENSE SWITCH 1)6,10
      6 PRINT 106
        IF(M-6)7,7,9
      7 DO 8 I=1,M
      8 TYPE 107, I
        TYPE 108
      9 EXECUTE PROCEDURE 300
C
C       CRACOVIAN ROOT
C
     10 CR(1)=SQRT(CR(1))
        DO 11 I=2,M1
     11 CR(I)=CR(I)/CR(1)
        J=2
     12 I=J
        K=I+(J-1)*(M2-J)/2
        K3=K
        I1=I-1
        DO 13 L=1,I1
        K1=I+(L-1)*(M2-L)/2
     13 CR(K)=CR(K)-CR(K1)**2
        CR(K)=SQRT(CR(K))
        IF(J-M1)14,17,14
     14 I=I+1
        K=K+1
        DO 15 L=1,I1
        I2=(L-1)*(M2-L)/2
        K1=I+I2
        K2=J+I2
     15 CR(K)=CR(K)-CR(K1)*CR(K2)
        CR(K)=CR(K)/CR(K3)
        IF(I-M1)14,16,14
     16 J=J+1
        GO TO 12
     17 IF(SENSE SWITCH 1)18,19
     18 PRINT 109
        EXECUTE PROCEDURE 300
```

```
C
C     CALCULATION OF UNKNOWNS
C
   19 I=M
   20 J=M
      I1=(I-1)*(M2-I)/2
      A=0.
   21 IF(J-I)22,23,22
   22 K=J+I1
      A=A+CR(K)*X(J)
      J=J-1
      GO TO 21
   23 K=M1+I1
      K1=I+I1
      X(I)=(-CR(K)-A)/CR(K1)
      IF(I-1)24,25,24
   24 I=I-1
      GO TO 20
   25 IF(SENSE SWITCH 2)26,28
   26 PRINT 112
      CONTROL 102
      T=M
      ECERO=CR(NX)/SQRT(T)
      DO 27 I=1,M
   27 PRINT 102,I,X(I)
      GO TO 39
C
C     INVERSE OF CRACOVIAN ROOT
C
   28 I=1
   29 J=I
      I1=(I-1)*(M2-I)/2
      K=I+I1
      CR(K)=1./CR(K)
      IF(I-M)30,33,30
   30 J=J+1
      K=K+1
      I2=J-1
      K3=J+(J-1)*(M2-J)/2
      A=0.
      DO 31 L=I,I2
      K1=J+(L-1)*(M2-L)/2
      K2=L+I1
   31 A=A-CR(K1)*CR(K2)
      CR(K)=A/CR(K3)
      IF(J-M)30,32,30
   32 I=I+1
      GO TO 29
   33 IF(SENSE SWITCH 1)34,35
   34 PRINT 110
      M1=M
      EXECUTE PROCEDURE 300
```

```
C
C       CALCULATION OF WEIGHT COEFFICIENTS OF UNKNOWNS
C
   35 L=1
      DO 37 I=1,M
      A=0.
      DO 36 J=1,M
      A=A+CR(L)**2
   36 L=L+1
      CR(L)=A
   37 L=L+1
      T=NOBS-M
      ECERO=CR(NX)/SQRT(T)
      PRINT 112
      TYPE 105
      M1=M+1
      K=M1
      DO 38 I=1,M
      EMX=ECERO*SQRT(CR(K))
      PRINT 102,I,X(I),EMX,CR(K)
      M1=M1-1
   38 K=K+M1
   39 PRINT 111,ECERO
      PAUSE
      GO TO 1
      BEGIN PROCEDURE 300
      IF(M-6)40,40,47
   40 L=1
      DO 45 I=1,M1
      DO 41 J=1,M1
      TYPE 101,CR(L)
   41 L=L+1
      IF(I-M1)42,46,42
   42 CONTROL 102
      IF(M-M1)44,43,44
   43 L=L+1
   44 DO 45 K=1,I
   45 TYPE 113
   46 RETURN 300
   47 L=1
      DO 50 I=1,M1
      DO 48 J=1,M1
      PRINT 101 ,CR(L)
   48 L=L+1
      IF(M-M1)50,49,50
   49 L=L+1
   50 CONTROL 102
      END PROCEDURE 300
```

```
100 FORMAT(2I4)
101 FORMAT(E11.4)
102 FORMAT(2HX(,I3,2H)=E11.4,2E15.4)
103 FORMAT(//9HUNKNOWNS=I4/13HOBSERVATIONS=I4//)
105 FORMAT(5X10HMEAN ERROR 5X10H Q COEFF. //)
106 FORMAT(/32HCOEFFICIENTS OF NORMAL EQUATIONS//)
107 FORMAT(4X2HX(,I2,3H ) )
108 FORMAT(3X7HC. TERM//)
109 FORMAT(/14HCRACOVIAN ROOT//)
110 FORMAT(/25HINVERSE OF CRACOVIAN ROOT//)
111 FORMAT(/19HMEAN SQUARE ERROR =E11.4//)
112 FORMAT(//10X8HUNKNOWNS)
113 FORMAT(11X)
    END
```

UNKNOWNS= 4
OBSERVATIONS= 8

COEFFICIENTS OF NORMAL EQUATIONS

X(1)	X(2)	X(3)	X(4)	C. TERM
.1200E 02	-.2000E 01	-.4000E 01	.0000E-50	-.2000E-01
	.6000E 01	.0000E-50	-.2000E 01	.5600E 00
		.2000E 02	-.7000E 01	.2020E 01
			.1700E 02	-.2560E 01
				.2176E 01

CRACOVIAN ROOT

.3464E 01	-.5773E 00	-.1154E 01	.0000E-50	-.5773E-02
	.2380E 01	-.2800E 00	-.8401E 00	.2338E 00
		.4311E 01	-.1678E 01	.4821E 00
			.3671E 01	-.4233E 00
				.1307E 01

INVERSE OF CRACOVIAN ROOT

.2886E 00	.7001E-01	.8186E-01	.5344E-01
	.4200E 00	.2728E-01	.1086E 00
		.2319E 00	.1060E 00
			.2723E 00

	UNKNOWNS	MEAN ERROR	Q COEFF.
X(1)=	-.3154E-01	.2044E 00	.9779E-01
X(2)=	-.6540E-01	.2842E 00	.1890E 00
X(3)=	-.6694E-01	.1667E 00	.6503E-01
X(4)=	.1153E 00	.1780E 00	.7419E-01

MEAN SQUARE ERROR = .6537E 00

MRI PLOTTER SUBROUTINES

by

David A. Bingham
Dean W. Lawrence

MRI PLOTTER SUBROUTINES

INTRODUCTION

This set of plotter subroutines is provided in the hope of eliminating much of the redundant effort that has apparently been expended in past uses of the IBM 1620's on-line plotter.

The set is built around modified versions of the IBM-provided PLOT and CHAR routines. The MRI versions of PLOT and CHAR offer the following features:

- Both can be used as LOCAL subroutines.
- Changing directions of lettering the CHAR routine takes less than 1/5 second.
- The PLOT routine can be reinitialized without having executed a CALL PLOT(7) statement.
- The PLOT routine can move the pen to any location without dropping the pen when the move is terminated.
- New graphs can be automatically started above, below, and to the left of the present graph (as well as to the right).
- Up to nine unique characters can be used to plot points instead of the standard + character.

The write-up on PLOT and CHAR is essentially that found in IBM publication No. C26-5841 with appropriate changes.

SUBROUTINE PLOT

The following definitions apply to the terms used in the description of this subroutine. All X and Y values are to be expressed in inches.

1. XMIN ≤ minimum X value to be plotted.
2. XMAX ≥ maximum X value to be plotted.
3. XL = the required physical length of the plot in the X direction.

4. XD = X increment to be indicated on plot outline = (XMAX - XMIN)/INC, where INC equals the number of increments desired.

5. YMIN $\leq$ minimum Y value to be plotted.

6. YMAX $\geq$ maximum Y value to be plotted.

7. YL = the required physical length of the plot in Y direction.

8. YD = Y increment to be indicated on plot outline = (YMAX - YMIN)/INC, where INC equals the number of increments desired.

9. IC = control integer which must have one of the following values: 1, 101, 201, 0, 9, 89, 90, 99, 7, 70, -7, -70. See "Use of Control Integer" under the specific Programming Procedure in question.

Choice of variable names for the above information is left to the discretion of the programmer, except for the restrictions that the control integer must be a fixed-point variable or constant and the rest be floating-point variables (with or without subscripts), constants or expressions. Array names must not be used.

The first time the subroutine is entered, scaling factors are calculated and a plotting outline or grid is drawn governed by the control integer IC. During subsequent entries into the subroutine, the X and Y coordinates of a point are scaled and the point is plotted governed by the control integer. After all points have been plotted, the subroutine is reinitialized for a new plot. In brief, the subroutine has three sections:

1. Framing and scaling

2. Point-to-point plotting

3. Reinitialization

Each of these subroutine sections has a set of control integers (IC) to govern the functions of the subroutine.

Framing and Scaling Section

In this section, two scaling constants are calculated and a plot outline or grid is drawn governed by the control integer IC. The subroutine calculates these scaling constants as follows:

$$CX = X \text{ scaling constant} = \frac{XL}{XMAX-XMIN}$$

$$CY = Y \text{ scaling constant} = \frac{YL}{YMAX-YMIN}$$

These scaling constants are then used to scale all X and Y values in the point-to-point plot section.

After the frame is drawn the pen is placed in the up status.

Use of Control Integer (IC)

One of the following three control integers must be used in this section:

IC = 1

IC = 101

IC = 201

When IC = 1 is used, scale constants are calculated and a plotting area outline is drawn (Fig. 1). When IC = 101 is used, scale constants are calculated and a grid over the plotting area is drawn (Fig. 2). When IC = 201 is used, scale constants are calculated, but neither a plotting area outline nor grid is drawn. (For efficient programming, set XD equal to XMAX-XMIN and YD equal to YMAX-YMIN, when IC = 201 is used.) The latter control integer is used when a reference frame is already available or when one is not required.

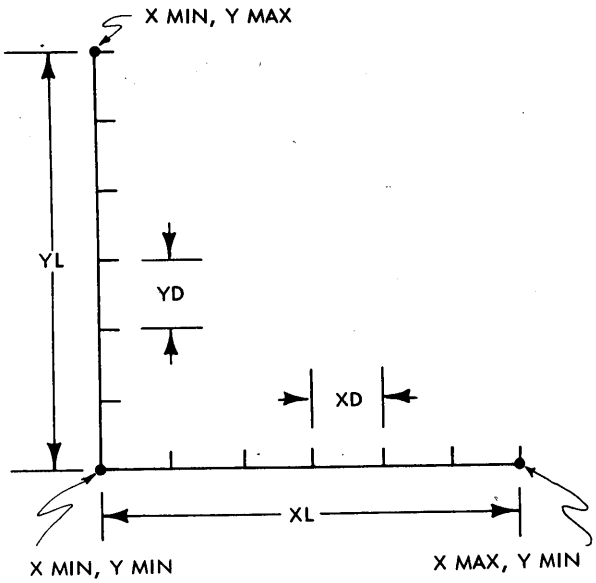


Fig. 1 - IC = 1

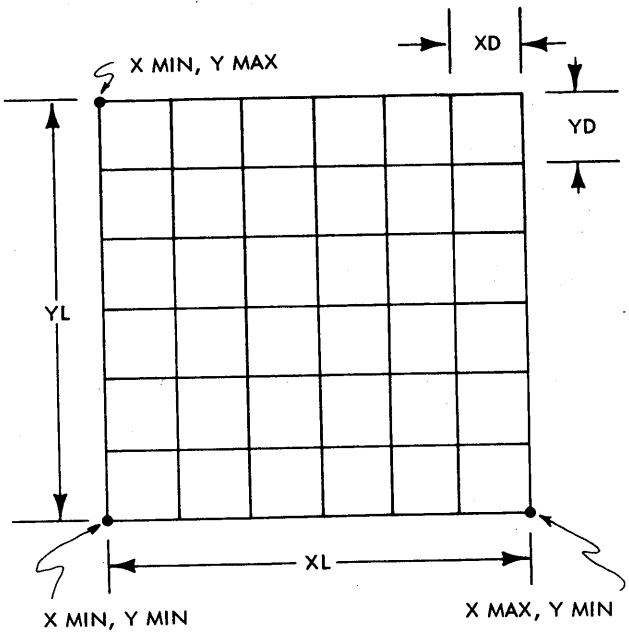


Fig. 2 - IC = 101

Point-Point Plot Section

CALL PLOT (IC, ICHAR, X, Y)

This section of the subroutine first scales a given point (X, Y) to a set of coordinates (X plot, Y plot) as follows:

$$X \text{ plot} = (X-XMIN) * CX$$

$$Y \text{ plot} = (Y-YMIN) * CY$$

where CX and CY are the scaling constants calculated in the framing and scaling section of the subroutine.

Use of Control Integer (IC)

One of the following two control integers must be used in this section:

NOTE: The pen always moves to the point to be plotted in the status in which it was left (up or down) from the previous call.

IC = 0

IC = 9

IC = 0 . When IC = 0 is used, the point (X, Y) is scaled, the pen moves to the resultant coordinate (X plot, Y plot) one of 9 characters is drawn through the point. The pen remains down after the character is drawn.

IC = 9 . When IC = 9 is used, the point (X, Y) is scaled, the pen moves to the resultant coordinate (X plot, Y plot) one of 9 characters is drawn through the point. The pen is placed in the up status after the character is drawn.

Use of Character Control Integer (ICHAR)

One of the following nine character control integers must be used:

POINT REPRESENTATION	VALUE OF ICHAR
.	1
+	2
o	3
□	4
▷	5
◇	6
×	7
∟	8
⁊	9

Fig. 3

Pen Movement Section

CALL PLOT (IC, X, Y)

This section of the subroutine scales a given point (X, Y) to (X plot, Y plot) as in the point-to-point section.

Use of the Control Integer (IC)

IC must assume one of the following three values:

IC = 89 . When IC = 89 is used, the point (X, Y) is scaled and the pen moves to the resultant coordinates (X plot, Y plot). The pen is commanded up when the point is reached.

IC = 90 . When IC = 90 is used, the point (X, Y) is scaled, and the pen moves to the resultant coordinate (X plot, Y plot). The pen is commanded down when the point to be plotted is reached.

IC = 99 . When IC = 99 is used, the pen is commanded up. No other action takes place.

NOTE: If the pen up or down status is altered manually or by some routine other than PLOT (for example, the annotation routine) IC = 99 should be executed prior to the resumption of point-to-point plotting. The reason for this lies in the fact that PLOT is designed to keep track of pen up or down status so that the pen does not have to be commanded to assume its present status. (One hundred ms are required for each pen up or down command.) By executing an IC = 99 call, as just described, pen up status is established physically and within the program.

Reinitialization Section

CALL PLOT (IC)

After all points have been plotted within a frame, PLOT can be called to automatically move the pen to a new position prior to starting a new frame. The following table shows the possible options available:

Value of IC	Position to Which Pen Will Be Moved
7	(XMAX + 3.0", YMIN)
70	(XMIN, YMAX + 3.")
- 7	(XMIN - (XL + 3."), YMIN)
-70	(XMIN, YMIN - (YL + 3."))

NOTE: PLOT does not have to be reinitialized with one of these calls. An alternative would be

CALL PLOT (89, X desired, Y desired)

CALL PLOT (1, ...)

These two statements would start a new frame at the point (X desired, Y desired).

PLOT PROGRAMMING PROCEDURE

Prior to calling the PLOT subroutine, the programmer may determine and place in storage the XMIN, XMAX, XL, XD, YMIN, YMAX, YL, YD, IC, and ICHAR information as previously defined.

Access to the PLOT Subroutine

The PLOT subroutine is accessed by the following statement:

```
CALL PLOT (IC, LIST)
```

where IC must be equal to 1, 101, 201, 0, 9, 89, 90, 99, 7, 70, -7, or -70. LIST depends upon the value of the control integer (IC) as follows:

If IC = 1, 101, or 201, LIST must contain XMIN, XMAX, XL, XD, YMIN, YMAX, YL, and YD as previously defined.

If IC = 0 or 9, LIST must contain ICHAR (the character control integer), X, and Y where X and Y are the coordinates of the point to be plotted.

If IC = 89 or 90, LIST must contain X and Y where X and Y are the coordinates of the point to be plotted.

If IC = 7, 70, -7, -70, or 99, LIST must not be used.

The actual names of the variables are unimportant as long as IC is a fixed-point variable or constant and LIST is made of floating-point variables, constants, or expressions, except that array names must not be used.

Because of the makeup of the subroutine, the first time it is entered the following statement must be used:

```
CALL PLOT (IC, XMIN, XMAX, XL, XD, YMIN, YMAX, YL, YD)
```

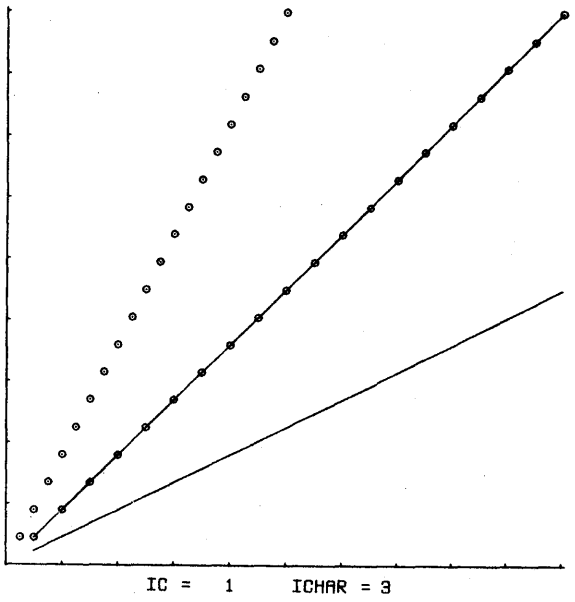
where IC must equal 1, 101, or 201.

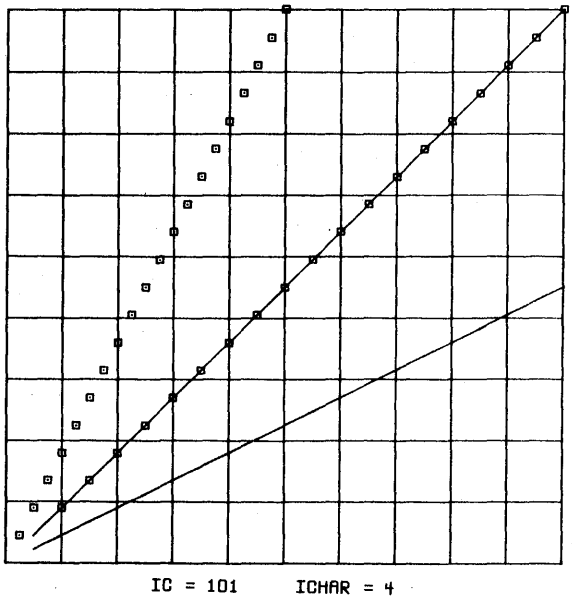
After the initial statement has been executed, the subroutine can be entered at any time by one of the following three statements:

```
CALL PLOT (IC, ICHAR, X, Y)

CALL PLOT (IC, X, Y)

CALL PLOT (IC)
```





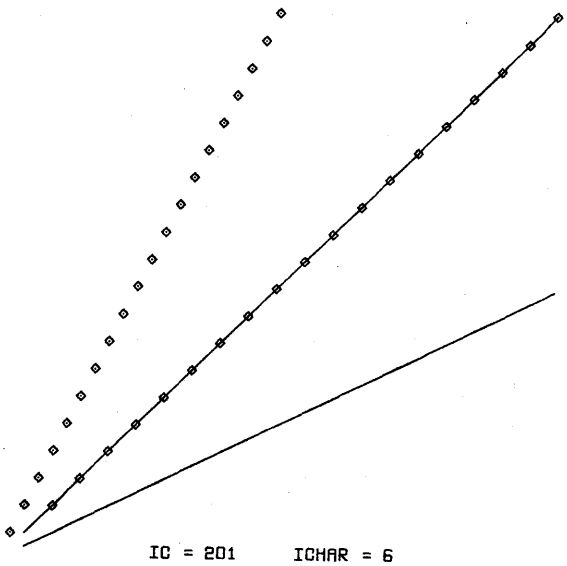


Fig. 4 - Plotter Options

SUBROUTINE CHAR

This relocatable subroutine is titled CHAR. As a subroutine, it allows a FORTRAN II or FORTRAN II-D program to plot letters, numbers and all FORTRAN special characters from information supplied by Format statements.

Program Description

The CHAR subroutine uses the typewriter I/O subroutine from the FORTRAN II or FORTRAN II-D Programming System. When the CHAR subroutine is called, it modifies the typewriter I/O routine to perform the lettering function and restores it when the function is complete. The position of the pen, at the time CHAR is called, is at the lower left-hand corner of the character to be drawn. Because the information to be plotted is obtained from Format statements furnished by the programmer, actual variable values can be plotted at object time.

Each character that is plotted is represented in the subroutine by a string of digits. These digits, when transmitted to the plotter, cause the proper pen and paper motion required to draw the desired character. The same string of digits is used for all character sizes and for horizontal or vertical displacement of the character. Horizontal or vertical placement of characters with respect to the X axis is accomplished internally by the subroutine. The conversion from horizontal placement to vertical placement or from vertical to horizontal, requires less than 1/5 second.

When all of the specified characters are drawn, the pen is returned to the starting position with the pen in the up status.

ABCDEFGHIJKLMNOPQRSTUVWXYZ

0123456789 =,$.(*)/+

.1 Inch Lettering

ABCDEFGHIJKLMNOPQRSTUVWXYZ

0123456789 =,$.(*)/+

.2 Inch Lettering

Fig. 5 - Character Set

Programming Procedures

Because of the design of the annotation subroutine, it must be accessed by the following two FORTRAN statements in the order shown.

CALL CHAR (N, SIZE, IHV, VAR_1, VAR_2--VAR_N)

FORMAT (S_1, S_2---S_N)

The arguments in the call statement are defined as follows:

N = A fixed-point variable or constant equal in value to the number of variables and/or constants (VAR) whose object time values are to be plotted as specified in the Format statement that follows the Call statement.

Example: If one variable (VAR_1) is to be plotted, N = 1.

SIZE = A floating-point variable or constant equal in value to the size (expressed in inches) of the characters to be plotted.

Example: If characters are to be drawn two-tenths of an inch high, SIZE = 0.2.

IHV = A fixed-point variable or constant. If IHV = 0, characters are drawn parallel to the X-axis, otherwise the characters are drawn parallel to the Y-axis.

VAR_N = Floating-point or fixed-point variables or constants whose values are drawn at object time. The Format specifications to which these variables are drawn must be specified in the Format statement following the Call statement.

The Format statement used with the Call statement supplies information to the subroutine as to what is to be drawn and under what Format specification. It is also used to calculate the correct return address from the subroutine. It is, therefore, mandatory that a Format statement follow the Call statement. The items in the Format statement are defined as follows:

S = A Format specification. The Format specification(s) may not be separated by a slash (/).

n = A statement number.

Program Examples

Problem: Draw an alphameric field, horizontally, using characters two-tenths of an inch high.

Solution:

```
CALL CHAR (0, 0.2, 0)

n FORMAT (6HX AXIS)
```

NOTE: Because there are no variables to be plotted, N, the first argument of the call is a fixed-point zero.

Problem: Draw an alphameric field, horizontally, and the variable (X) using characters one-tenth of an inch high.

Solution:

```
CALL CHAR (1, 0.1, 0, X)

n FORMAT (14H VALUE OF X IS, F6.3)
```

NOTE: Because one variable is to be plotted, the first argument of the Call statement is a fixed point 1.

Problem: Draw an alphameric field, vertically, and three variables (X, Y, Z) using characters one-tenth of an inch high.

Solution:

```
CALL CHAR (3, 0.1, 1, X, Y, Z)

n FORMAT (21HXYZ HAVE THE VALUES, 3F 6.3)
```

Problem: Draw three variables (X, Y, Z) using characters one-tenth of an inch high.

Solution:

```
CALL CHAR (3, 0.1, 1, X, Y, Z)

n FORMAT (3F 8.4)
```

SUBROUTINE FRAME (XMIN, XMAX, XL, XD, YMIN, YMAX, YL, YD)

This subroutine draws a complete rectangular frame with axis markings ("tics") along all sides. The arguments are those defined in the PLOT subroutine with the following exception:

If XD = 0., the X axis is marked off in $\log_{10}$ increments. In this case, XMIN and XMAX are assumed to be log quantities. For example, if XMIN = 1., XMAX = 4., and XD = 0., then three complete cycles of $\log_{10}$ marks will be drawn on the X axis (both top and bottom). If YD = 0., the same remarks hold true for the Y axis. If both XD and YD are zero, then $\log_{10}$ marks will be drawn on both axes.

NOTE: FRAME initializes the PLOT routine with the following statement:

```
CALL PLOT (201, XMIN, XMAX, XL, XMAX-XMIN, YMIN, YMAX, YL, YMAX-YMIN)
```

FRAME uses about 16,994 positions of core storage. The following illustrations are examples of the use of FRAME:

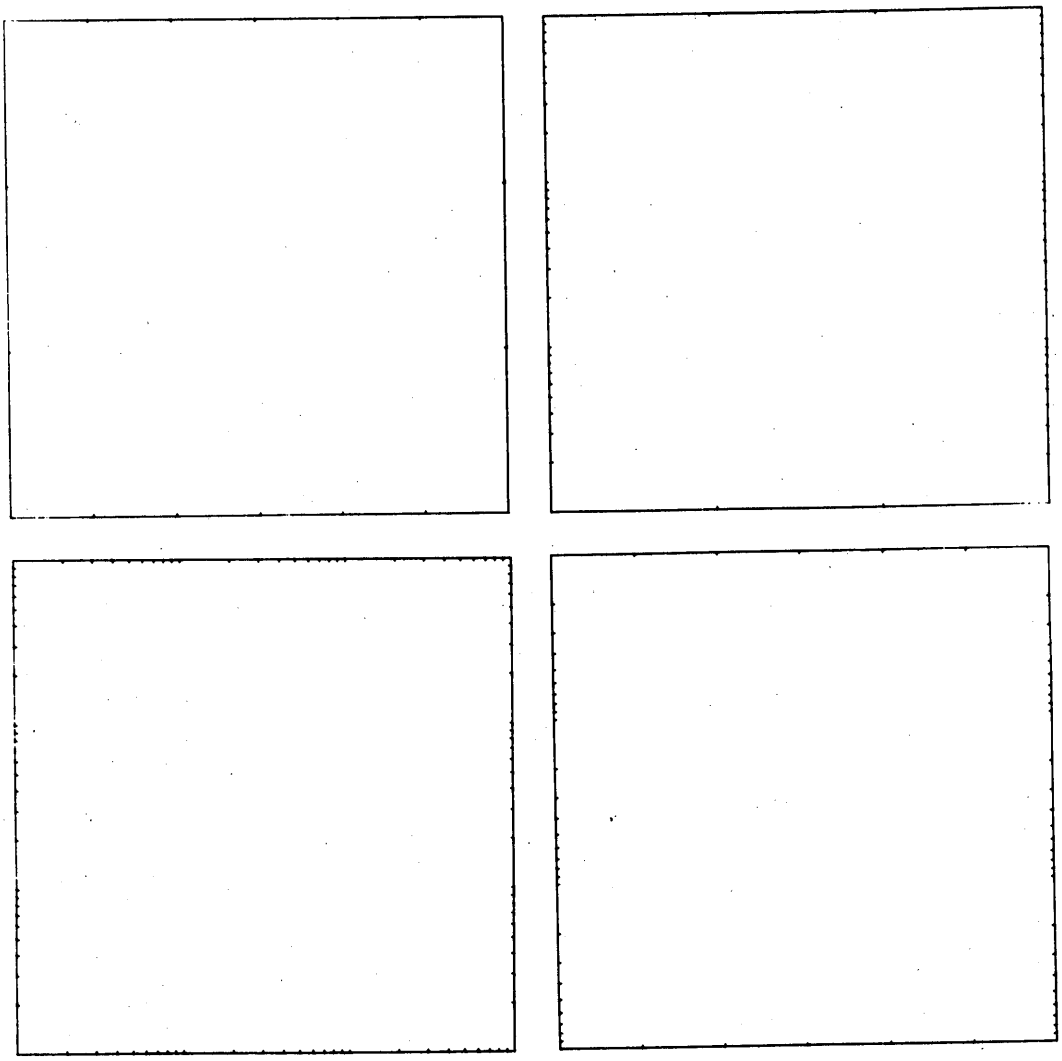


Fig. 6 -

```
C       ---- Framing demo -- Start in upper left corner, draw four types
C       of frames.  Sequence is clockwise.
C
        CALL FRAME (1.,4.,6.,1.,1.,4.,6.,.5)
        CALL PLOT(7)
        CALL FRAME (1.,4.,6.,1.,1.,4.,6.,0.)
        CALL PLOT(-70)
        CALL FRAME (1.,4.,6.,0.,1.,4.,6.,.5)
        CALL PLOT(-7)
        CALL FRAME (1.,4.,6.,0.,1.,4.,6.,0.)
        PAUSE
        END
```

SUBROUTINE GRAPH (I, J, IPOINT, XINCH, YINCH, X, FX)

This subroutine is provided to graph the points (X(K), FX(K)) and draw curvilinear lines joining the points. The arguments are defined as:

X(K) = A single-subscripted array of the abscissa (X-axis) values

FX(K) = A single-subscripted array of ordinate values

I = The subscript of the first point to be plotted

J = The subscript of the last point to be plotted

IPOINT = The digit that governs which point character is to be used (see character control integer in PLOT arguments). If IPOINT is zero, the actual points will not be plotted.

XINCH = The value of one physical inch in terms of X-axis measure

$$XINCH = \frac{XMAX-XMIN}{XL}$$ in terms of the PLOT subroutine arguments.

YINCH = The value of one physical inch on the Y-axis.

GRAPH uses the second order Lagrangian polynomial over X(K), X(K+1), and X(K+2) to draw the curve between X(K) and X(K+1). (When K+2 = J, the polynomial is also used to draw the curve between X(K+1) and X(J).) The curve is composed of short line segments less than 0.1 inch long. The actual length of these segments is a variable that depends on the physical distance from (X(K), FX(K)) to (X(K+1), FX(K+1)).

The following assumptions are made for GRAPH: (1) The points are stored in either ascending or descending values of X. (2) No two successive values of X are equal.

GRAPH uses about 14,374 positions of core.

The following illustrations show the two types of plots available when GRAPH is used.

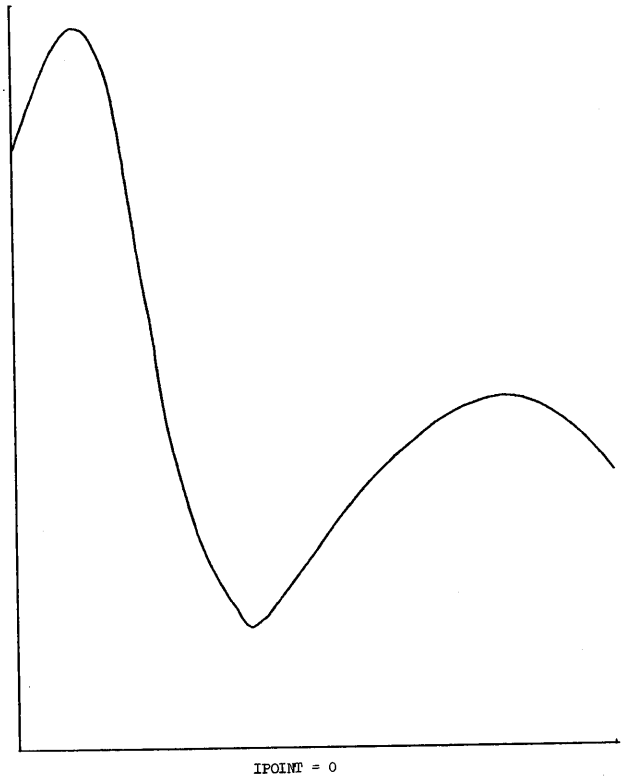


Fig. 7

SUBROUTINE LABEL (SIZE, XMAX, XMIN, XL, XD, YMIN, YMAX, YL, YD)

SUBROUTINE LABELX (Same Arguments)

SUBROUTINE LABLE (Same Arguments)

SUBROUTINE LABLEX (Same Arguments)

The purpose of these four subroutines is to place labels along the X and Y axes drawn by the initializing call of the PLOT subroutine, i.e.,

(1) CALL PLOT (IC, XMIN, XMAX, XL, XD, YMIN, YMAX, YL, YD)

where IC = 1 or 101.

The arguments are the same for all four subroutines. Specifically,

SIZE = The value of the character size expressed in inches (e.g., SIZE = .2 means all characters will be .2 of an inch in height)

XMIN, XMAX, XL, YMIN, YMAX, YL = Must be exactly the same as used in (1)

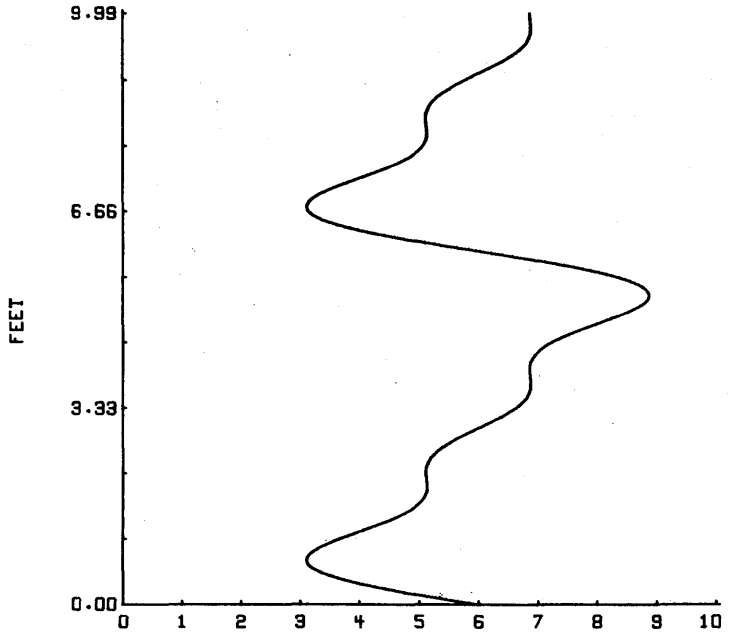
XD, YD = The same as used in (1) or integer multiples* of that value

All four subroutines will place numerical values adjacent to the "tic" marks on each axis. In addition LABLE and LABLEX read two cards; the first is used as an X-axis heading and the second as the Y-axis heading. (Column 40 is assumed to be the midpoint of the axis.) The following chart shows the exact distinctions of the four routines.

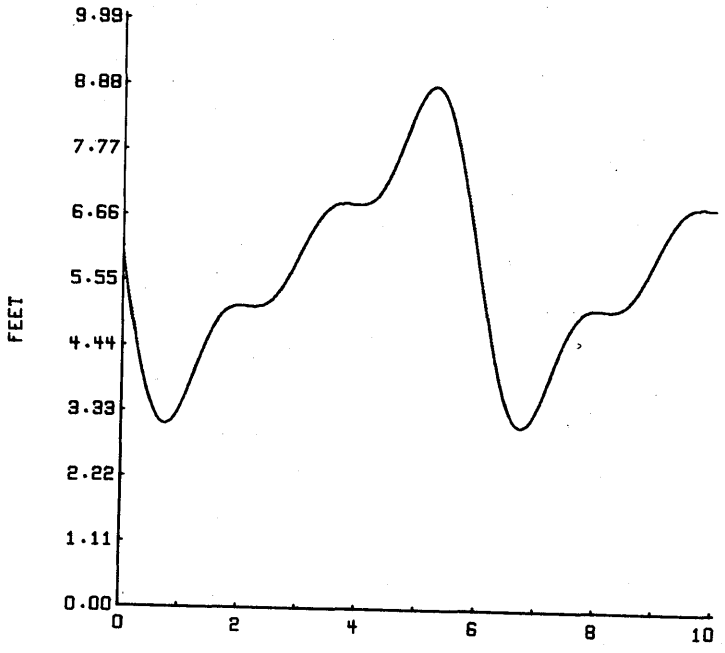
* If XD is the value used when PLOT is initialized, and 3.*XD is used as this argument, then every third "tic" mark on the X axis will be labeled with a value.

Name	Range of Values	Format of Representation	Axis Headings	Core Required
LABEL	-999 to 9999	I4	No	1402
LABELX	Any	MAX* ≥ 10,000 ; E8.1 10 ≤ MAX < 10,000 ; I5 .1 ≤ MAX < 10 ; F5.2 MAX < .1 ; E8.1	No	2550
LABLE	Same as LABEL		Yes	2314
LABLEX	Same as LABELX		Yes	3472

* MAX = is the maximum of |XMAX| and |XMIN| (or of |YMAX| and |YMIN|); e.g., if XMIN = -10 and XMAX is 0 then MAX = 10 and I5 format is used.




```
CALL PLOT (1,0.,10., 5.,1.,0.,9.99, 5.,1.11)
CALL LABLEX (.1,0.,10., 5.,1.,0.,9.99, 5.,3.33)
```




```
CALL PLOT (1,0.,10., 5.,1.,0.,9.99, 5.,1.11)
CALL LABLEX (.1,0.,10., 5.,2.,0.,9.99, 5.,1.11)
```

Fig. 8

SUBROUTINE QUADRG (N, X, Y, A, B, C)

The subroutine QUADRG computes A, B, and C in the second-order regression formula:

$$Y = A + BX + CX^2$$

X = The single-subscripted array of independent variable values

Y = The single-subscripted array of dependent variable values

N = The number of (X, Y) values

QUADRG requires about 2300 core positions and the single subscript library subroutine.

This routine will be particularly useful in plotter applications. The regression curve can be plotted by the following sequence of statements:

```
XJ = J*
DX = (X(N) - X(1))/XJ
X1 = X(1)
CALL PLOT (99)
CALL PLOT (90, X1, A+B*X1+C*X1*X1)
DO 10 I = 1, J
X1 = X1 + DX
10 CALL PLOT (90, X1, A+B*X1*C*X1*X1)
```

* Where J is some suitable number that depends on the degree of curvature and the physical distance between X(1) and X(N) or the plot.

SUBROUTINE LINREG (N, X, Y, A, B)

LINREG is a subroutine that computes the y-intercept, A, and slope, B, of the linear regression line fitted to the points (X(I), Y(I)), I = 1,2..., N. (That is, the linear least squares fit to a set of data points.)

X, the independent variable, and Y, must be single-subscripted in the calling program.

The running time of the subroutine is dependent on N, the number of data points. Roughly, it computes about one second for every 15 data points.

This subroutine will be particularly useful in plotter applications. The regression line can be plotted by the following sequence of statements:

```
CALL PLOT (99)

CALL PLOT (90, X(1), A + B*X(1))

CALL PLOT (90, X(N), A + B*X(N))
```

LINREG requires 1056 core positions and the single subscript library subroutine.