PROCEEDINGS

of the Meeting

of the Western Region

of Common

Denver Hilton Hotel

Denver, Colorado

July 6, 7, 8, 1966

AGENDA

Western Region Summer Meeting of Common
Denver Hilton Hotel, Denver, Colorado
July 6, 7, 8, 1966

Wednesday, July 6

General 1130 Session

9:00 a.m.	Welcome IBM Announcements
9:45 a.m.	1130 Monitor System
11:15 a.m.	1130 Programming Techniques
1:30 p.m.	Program for Optical Systems Design
3:30 p.m.	1130 Project Control Systems
4:00 p.m.	1130 General Discussion Users Panel

Parallel 1130 Sessions

11:15 a.m.	Graphic Report Generator
1:30 p.m.	1130 STRESS - Structural Engineering System Solver

Evening Session

7:00 p.m.	New Users Session
7:30 p.m.	Sound off Session

Thursday, July 7

General Session

9:00 a.m.	1130 Scientific Subroutine Package
10:30 a.m.	Solution of the Induction Well Logging Response Equation, G. Copland, Halliburton

Thursday, July 7 - continued

General Session - cont.

11:00 a.m.	The Direct and Cyclic Jacobi Methods with Fadeeva Correction Algorithm for a Real Matrix H. Fettis - J. Caslin, Wright Patterson AFB
11:50 a.m.	Luncheon Speaker - Ed Schwarz, Product Administrator - 1130 System
1:30 p.m.	Autospot Post Processor Development D. Oliver, ACF Industries
2:00 p.m.	Design of ME-LTV Evaporators used in Sea Water Conversion D. Kays, Stearns - Roger Corp.
2:20 p.m.	Development of a Public Works Engineering D.P. Center J. Hunter, L.A. County Dept. of Engineering
3:30 p.m.	FLBSPS AND STOVE K. Jones, Colorado Dept. of Highways

Parallel Session

Education

10:30 a.m.	University of Mississippi Test Scoring Program R. Ross, U. of Mississippi
11:00 a.m.	Student Scheduling on the 1620 G. Crumley, The Citadel
11:30 a.m.	Decoding and Use of the Special 1230 Marking Code on 1620 Without Column Binary C. Stallings, Jr., U. of New Mexico
1:30 p.m.	Whitman College Registration and Grading System G. Purcell, Whitman College

Thursday, July 7 - continued

Parallel Session - cont.

Education

2:00 p.m.	Whitman College Budget Forecasting G. Purcell, Whitman College
2:30 p.m.	University of Mississippi Floating Point Subroutines R. Ross, U. of Mississippi
3:30 p.m.	Austin College SPS - Improved D. Musser, Austin College
4:00 p.m.	Student D.P. System at Christian Brothers College Employing an IBM 1620 J. Wegener, Christian Brothers College

Structural

1:45 p.m.	Matrix Structural Analysis E. Cook, Wichita State University
2:15 p.m.	Design of Heavy Multi-Story Rigid Frame Industrial Frame Structures V. Arndt, Stearns Roger Corp.
3:30 p.m.	General Discussion Period (Structural and Civil)

Friday, July 8

General Session

9:00 a.m.	Library, Magazines and a Computer G. Ahlborn, U. of California, Berkeley
9:30 a.m.	Accurate Solution of Systems of Linear Equations D. Musser, Austin College
10:30 a.m.	How a Programmer Should Write a Program for an Engineer L. Mahoney, R. W. Beck Associates

FRANK AANERUD
STEARNS-ROGER CORP.
P.O. BOX 5888
DENVER, COLORADO 80217

C. WILLIAM ADE
MISSISSIPPI RIVER TRANS. CORP.
9900 CLAYTON ROAD
ST LOUIS MISSOURI

MR. GALE AHLBORN
UNIVERSITY OF CALIFORNIA, I.T.T.E.
1301 SOUTH 46TH STREET
RICHMOND, CALIFORNIA 94804

JERRY K. AIKAWA, M.D.
UNIVERSITY OF COLORADO MEDICAL CENTER
4200 EAST 9TH AVENUE
DENVER, COLORADO 80220

(MRS) BETTY ALEXANDER
UNIVERSITY OF SASKATCHEWAN
REGINA CAMPUS
REGINA, SASKATCHEWAN, CANADA

DAVE ALEXANDER
I. B. M. CORP.
2255 ALBERT STREET
REGINA, SASKATCHEWAN, CANADA

VERN ARNDT
STEARNS-ROGER CORP.
P.O. BOX 5888
DENVER, COLORADO 80217

EUGENE BARDACH
I.B.M. CORP.
245 MARQUETTE
MINNEAPOLIS, MINN.

GOLLIN BELL
PAFFORD AND ASSOCIATES
2898 ROWENA AVENUE
LOS ANGELES, CALIF.

RICHARD M. BELT
HAWAIIAN ELECTRIC CO., INC.
P.O. BOX 2750
HONOLULA, HAWAII

GUY E. BENNETT
I.B.M. CORP.
MONTEREY AND COTTLE ROADS
SAN JOSE, CALIF.

PAUL A. BICKFORD
O.S.U. TECH. INSTITUTE
1900 N.W. 10TH STREET
OKLAHOMA CITY, OKLA.

LT1 RICHARD J. BOLDUC
US ARMY AIR DEFENSE BOARD
TEST SUPPORT DIVISION
FORT BLISS, TEXAS

JAMES N. BOLES
UNIVERSITY OF CALIFORNIA
DEPT. OF AGRICULTURAL ECONOMICS
207 GIANNINI HALL
BERKELEY, CALIF.

CHARLES B. BRAGASSA
MEDICAL COLLEGE OF GEORGIA
1459 GWINETT STREET
AUGUSTA, GEORGIA

ROBERT L. BRIGHT
HAROLD HOSKINS AND ASSOC. INC.
1630 QUE STREET
LINCOLN NEBRASKA 68508

RICHARD F. BROOKS
MONSANTO CO.
1700 S 2ND STREET
ST LOUIS, MISSOURI

WILLIAM F. BURGGRABE, JR.
NOOTER CORPORATION
1400 SOUTH 3RD STREET
ST. LOUIS, MISSOURI

BRUCE A. BURNS
U. S. AIR FORCE
ENT AFB COLO.
COLORADO SPRINGS, COLORADO

LESTER R. BURRELL
CONSTRUCTION ESTIMATORS COMPANY, INC.
557 SOUTH SECOND WEST
SALT LAKE CITY, UTAH 84101

BOB BUTLER
I.B.M. CORP.
777 GRANT ST.
DENVER, COLORADO

STEPHEN W. BUTLER
J. M. HUBER CORPORATION
P.O. BOX 831
BORGER, TEXAS 79007

RICHARD M. CARTER
U. OF ILLINOIS
CENTER FOR THE STUDY OF MED. EDUCATION
901 S WOLCOTT
CHICAGO, ILLINOIS

JAMES C. CASLIN
AEROSPACE RESEARCH LABORATORIES (ARM)
WRIGHT-PATTERSON AIR FORCE BASE
OHIO 45433

N. R. CARSON
R. W. BECK AND ASSOCIATES
800 WESTERN FEDERAL SAVINGS BUILDING
DENVER, COLORADO 80202

JERE E. CHRISPENS
LOMA LINDA UNIVERSITY
SCIENTIFIC COMPUTATION FACILITY
LOMA LINDA, CALIFORNIA

DICK COLEMAN
AERO COMMANDER
MAX WESTHIMER FIELD
NORMAN, OKLAHOMA

EVERETT L. COOK
WICHITA STATE UNIV.
WICHITA, KANSAS

GEORGE V. COPLAND
HALLIBURTON CO.
ERD DEPT.
DUNCAN, OKLA.

JOHN H. CORNISH
DEPT. OF COMMERCE, ESSA
U.S. WEATHER BUREAU, RIVER FORCAST CENTER
9 EAST 4TH BLDG. ROOM 817
TULSA, OKLAHOMA

JAMES CRABTREE JR.
SUNDSTRAND AVIATION
2421 ELEVENTH STREET
ROCKFORD, ILLINOIS

T. J. CROTTY
I.B.M. CORP.
618 S MICHIGAN AVENUE
CHICAGO, ILLINOIS

WARD CROWLEY
UNIVERSITY OF IDAHO
COMPUTER CENTER
MOSCOW, IDAHO

GEORGE L. CRUMLEY
THE CITADEL
CHARLESTON, SOUTH CAROLINA 29409

P. R. CUNNINGHAM
R. W. BECK AND ASSOCIATES
800 WESTERN FEDERAL SAVINGS BUILDING
DENVER, COLORADO 80202

JIM DAVIDSON
STEARNS-ROGER CORP.
P.O. BOX 5888
DENVER, COLORADO 80217

JOHN C. DAY
I.B.M. CORP.
3424 WILSHIRE BLVD.
LOS ANGELES, CALIF.

GEORGE B. DENISON
FAIRBANKS MORSE AND CO.
LAWTON AVENUE
BELOIT, WISCONSIN

JOHN DENSEM
U. S. COAST GUARD
1300 E STREET N. W.
WASHINGTON, D. C.

RICHARD J. DERKS
EATON YALE AND TOWNE, INC.
DYNAMATIC DIV.
3122 - 14TH AVENUE
KENOSHA, WISCONSIN

LAURENT C. DESCHAMPS
U. S. COAST GUARD
1300 E STREET N. W.
WASHINGTON, D. C.

ROBERT C. DILLON
UNIVERSITY OF ARIZONA
TUCSON, ARIZONA

MRS. WM. C. DIMOND
I.B.M. CORP.
3800 LINDELL BLVD.
ST. LOUIS, MISSOURI

MARILYN L. DOIG
COLORADO STATE UNIVERSITY
FORT COLLINS, COLORADO

JOHN DUCKWORTH
I.B.M. CORP.
2605 E. PLATTE
COLORADO SPRINGS, COLORADO

T. J. DUESTERBERG
ANDERSEN, KOERWITZ AND HAWES, INC.
2701 ALCOTT STREET
SUITE 463
DENVER, COLORADO

BILL DUNCAN
STEARNS-ROGER CORP.
P.O. BOX 5888
DENVER, COLORADO 80217

DAVID A. DUNSMORE
OHIO RIVER VALLEY WATER SANITATION COMM
414 WALNUT STREET
CINCINNATI, OHIO 45202

DAVE DYE
I.B.M. CORP.
WHITE PLAINS, NEW YORK

MAJOR R. B. EDDINGTON, USAF
USAF - INSTITUTE OF TECHNOLOGY, DET NO. 5
MALMSTROM, AFB
GREAT FALLS, MONTANA

WALTER G. ELWELL
NEBRASKA WESLEYAN UNIV.
LINCOLN, NEBRASKA 68504

E. E. EVANS
MARTIN COMPANY
P. O. 179, MAIL NO. A-6641
DENVER, COLORADO

RICHARD FANSON
WHIRLPOOL CORP.
LAUNDRY ENG. DIV.
ST. JOSEPH, MICHIGAN

LLOYD W. FARNSWORTH
DEPARTMENT OF HIGHWAYS
STATE OF COLORADO
4201 E. ARKANSAS AVENUE
DENVER, COLORADO 80222

WILLIAM E. A. FENNEL
COMPUTER SERVICES INC.
850 RICHARDS STREET
HONOLULU, HAWAII 96813

JAMES B. FLINT
FALCON RESEARCH AND DEVELOPMENT CO.
1441 OGDEN STREET
DENVER, COLORADO 80218

JOHN P. FORD
EASTERN WASHINGTON STATE COLLEGE
CHENEY, WASHINGTON 99004

IRVINE H. FORKNER
METROPOLITAN STATE COLLEGE
250 WEST 14TH AVENUE
DENVER, COLORADO 80204

GUY A. GALLAWAY
NATIONAL CENTER FOR ATMOS. RESEARCH
SACRAMENTO PEAK OBSERVATORY
SUNSPOT, NEW MEXICO 88349

DONALD S. GARDNER
GENERAL FOODS
555 S. BROADWAY
TARRYTOWN, NEW YORK

H. MICHAEL GARGANO
U.S. PUBLIC HEALTH SERVICE
DIVISION OF RADIOLOGICAL HEALTH
1901 CHAPMAN AVENUE
ROCKVILLE, MARYLAND 20852

JACK D. GILLUM
JACK D. GILLUM AND ASSOCIATES
2049 BROADWAY
BOULDER, COLORADO

GEORGE W. GLADFELTER
S. DAKOTA SCHOOL OF MINES AND TECH
RAPID CITY, S. DAKOTA 57701

JOHN M. GOODE
HALLIBURTON COMPANY
DUNCAN, OKLAHOMA

W. C. GRAY
WAGNER ELECTRIC CORP.
11444 LACKLAND ROAD
CREVE COEUR, MISSOURI 63141

WAYNE H. GRIFFIN
GREAT CANADIAN OIL SANDS LTD.
500 ROYAL BANK BLDG.
EDMONTON, ALBERTA, CANADA

G. W. GUESCH
I. B. M. CORP.
MONTEREY AND COTTLE ROAD
SAN JOSE, CALIF.

KENNETH L. HARBAUM
THE PROCTER AND GAMBLE CO.
MIAMI VALLEY LABORATORIES
P.O. BOX 39175
CINCINNATI 39, OHIO

REGINALD T. HARLING
AIR FORCE INSTITUTE OF TECHNOLOGY
WRIGHT-PATTERSON AIR FORCE BASE, OHIO

DAVID R. HARRIS
UTAH STATE UNIVERSITY
DEPT. OF APPLIED STATISTICS
AND COMPUTING SCIENCE
LOGAN, UTAH 84321

J. J. HAYDEN
ACF INDUSTRIES, INC. ALBUQUERQUE DIV
336 WOODWARD ROAD S. E.
ALBUQUERQUE, NEW MEXICO

H. CURTIS HEACOX
UNI. OF WISCONSIN SPACE ASTRONOMY LAB.
35 NORTH PARK STREET
MADISON, WISCONSIN

DONALD R. HEIM
I.B.M. CORP.
1111 CONNECTICUTT AVE.
WASHINGTON, D.C.

DURWOOD HENDERSON
WEST TEXAS STATE UNIVERSITY
BOX 552, W. T. STATION,
CANYON, TEXAS

R. N. HERRING
BEECH AIRCRAFT CORP.
P. O. BOX 631
BOULDER, COLORADO

DR. A.A.J. HOFFMAN
TEXAS CHRISTIAN UNIVERSITY
BOX 30030A, TCU STATION
FORT WORTH, TEXAS 76129

E. ROGER HOFFMAN
MARTIN COMPANY
P. O. 179, MAIL NO. A-6641
DENVER, COLORADO

MARTHA HOLLENBECK
I.B.M. CORP.
340 MARKET STREET
SAN FRANCISCO, CALIFORNIA

JAMES W. HUNTER
LOS ANGELES COUNTY, DEPT. OF CO. ENGR.
108 W. 2ND STREET
LOS ANGELES, CALIFORNIA

LARRY D. JACKSON
COLORADO STATE UNIVERSITY
FORT COLLINS, COLORADO

DONALD L. JOHNSON
BUTLER MFG. CO.
7400 E. 14TH STREET
KANSAS CITY, MISSOURI

ARLIN JAMES
UNIVERSITY OF COLORADO MEDICAL CENTER
4200 EAST 9TH AVENUE
DENVER, COLORADO 80220

CHARLES N. JOLLIFFE
E. I. DUPONT
P.O. BOX 89
CIRCLEVILLE, OHIO

KENNETH W. JONES
DEPARTMENT OF HIGHWAYS
STATE OF COLORADO
4201 E. ARKANSAS AVE.
DENVER, COLORADO 80222

ED JURACEK
STEARNS-ROGER CORP.
P.O. BOX 5888
DENVER, COLORADO 80217

JOSEPH J. KALASZ
U. S. ARMY MISSILE COMMAND
REDSTONE ARSENAL, ALABAMA

R. T. KARSIAN
I.B.M. CORP.
777 GRANT STREET
DENVER, COLORADO

DAVE KAYS
STEARNS-ROGER CORP.
P.O. BOX 5888
DENVER, COLORADO 80217

H. B. KERR
TENN. TECH.
BOX 21A TTU
COOKEVILLE, TENNESSEE

WM. DALE KING
NATIONAL CENTER FOR ATMOS. RESEARCH
HIGH ALTITUDE OBSERVATORY
900 24TH STREET
BOULDER, COLORADO

K. A. KISSLING
PANHANDLE EASTERN PIPE LINE CO.
3444 BROADWAY
KANSAS CITY, MISSOURI

DONALD R. KLEIN
VILLANOVA UNIVERSITY COMPUTING CENTER
VILLANOVA UNIVERSITY
VILLANOVA, PENNSYLVANIA 19010

DARREL E. KNAUS
INTERNATIONAL BUSINESS MACHINES
2640 CANAL
NEW ORLEANS, LOUISIANA

RAYMOND G. KNEIP
PALO ALTO UNIFIED SCHOOL DIST.
25 CHURCHILL AVENUE
PALO ALTO, CALIF.

L. E. KRAMER
R. W. BECK AND ASSOCIATES
800 WESTERN FEDERAL SAVINGS BUILDING
DENVER, COLORADO 80202

JERYL W. LAFON
U.S. ARMY ENGINEER DIST. ALBUQUERQUE
P.O. BOX 1580
ALBUQUERQUE, NEW MEXICO 87103

WALLACE W. LAMOREUX
U.S. WEATHER BUREAU
4880 MACARTHUR BLVD. N.W.
WASHINGTON, D.C.

MARK LANDWEHR
I.B.M. CORP.
SAN JOSE, CALIFORNIA

VICTOR E. LATTIN
CENTRALIA COLLEGE
P.O. BOX 639
CENTRALIA, WASHINGTON

JOHN F. LAUER
COLORADO COMPUTING CORP.
BOX 38
BOULDER, COLORADO

GENE LESTER
I.B.M. CORP.
SAN JOSE, CALIF.

DAVID L. LORD
HALLIBURTON COMPANY
TECHNICAL CENTER
DUNCAN, OKLAHOMA

R. K. LOUDEN
I B M
SAN JOSE, CALIFORNIA

JENE Y. LOUIS
LONG ISLAND LIGHTING COMPANY
175 OLD COUNTRY ROAD
HICKSVILLE, NEW YORK

R. BRUCE MACMULLIN
WESTERN SUPPLY CO.
BOX 1888
TULSA, OKLAHOMA

DAVID MACURDY
FALK JORGENSEN CONSULTING ENGRS., INC.
1240 W. BAYAUD AVENUE
DENVER, COLORADO

HARRY MAH
GENERAL FOODS CORP.
555 S BROADWAY
TARRYTOWN, NEW YORK

L. F. MAHONEY
R. W. BECK AND ASSOCIATES
800 WESTERN FEDERAL SAVINGS BUILDING
DENVER, COLORADO 80202

PAUL MANIKOWSKI
I.B.M. CORP.
3424 WILSHIRE BLVD.
LOS ANGELES, CALIF.

RAYMOND P. MANYIK
U. S. AIR FORCE
ENT AFB COLO.
COLORADO SPRINGS, COLORADO

E. L. MATTHEWS
I.B.M. CORP.
MONTEREY AND COTTLE ROADS
SAN JOSE, CALIF.

CHARLES E. MAUDLIN, JR.
UNIV. OF OKLAHOMA
NORMAN, OKLAHOMA

TED MCKENNA
SUNDSTRAND AVIATION - DENVER
2480 W. 70TH AVENUE
DENVER, COLORADO

O. F. MERKLINGHAUS
I.B.M. CORP.
4843 BELA DRIVE
SAN JOSE, CALIF.

TERREL L. MIEDANER
UNI. OF WISCONSIN SPACE ASTRONOMY LAB.
35 NORTH PARK STREET
MADISON, WISCONSIN

STANLEY R. MILLER
U.S. NAVAL ORDNANCE TEST STATION
CODES 45301, 4535
CHINA LAKE, CALIF.

DARYL MONROE
MESA COLLEGE
GRAND JUNCTION, COLORADO

GEORGE MOORE
SUNDSTRAND AVIATION - DENVER
2480 W. 70TH AVENUE
DENVER, COLORADO

RICHARD W. MURRAY
MOLONEY ELECTRIC CO.
5390 BIRCHER BLVD.
ST. LOUIS, MISSOURI 63120

DAVID R. MUSSER
AUSTIN COLLEGE
SHERMAN, TEXAS

MARTHA MUSSER
WADLEY RESEARCH INSTITUTE
3600 GASTON AVENUE
DALLAS, TEXAS

DON MYERS
COLORADO STATE COLLEGE
BUREAU OF RESEARCH SERVICES
GREELEY, COLORADO 80631

JOHN MYERS
TEKTRONIX, INC.
P.O. BOX 500
BEAVERTON, OREGON

DAVID S. NELSON
LE TOURNEAU WESTINGHOUSE
2301 NE ADAMS STREET
PEORIA, ILLINOIS

VICTOR L. NOBLITT
I.B.M. CORP.
777 GRANT STREET
DENVER, COLORADO

BOYD C. NORRIS
U. S. BUREAU OF RECLAMATION
P. O. BOX 2511
SACRAMENTO, CALIFORNIA

LARRY G. NOTHWANG
WESTERN ENGINEERING CO
1445 WILLAMETTE STREET
EUGENE, OREGON

WARREN W. OSHEL
U.S. NAVAL ORDNANCE TEST STATION
CODES 45301, 4535
CHINA LAKE, CALIF.

H. C. PETERSON
THE NEW JERSEY ZINC CO.
PALMERTON, PENNSYLVANIA

MELVIN L. PIERCE
ARLINGTON STATE COLLEGE
2ND AND COLLEGE STREET
ARLINGTON, TEXAS 76010

EDWARD PINFIELD
UNIVERSITY OF COLORADO MEDICAL CENTER
4200 EAST 9TH AVENUE
DENVER, COLORADO 80220

WILLIAM C. PIQUETTE
OTERO JUNIOR COLLEGE
18TH AND COLORADO
LA JUNTA, COLORADO

ROBERT PLANTE
WHIRLPOOL CORP.
LAUNDRY ENG. DIV.
ST. JOSEPH, MICHIGAN

ALBERTA PLYM
UNIVERSITY OF COLORADO MEDICAL CENTER
4200 EAST 9TH AVENUE
DENVER, COLORADO 80220

REV. JOSEPH B. POMEROY S. J.
COLLEGE OF THE HOLY CROSS
WORCESTER, MASSACHUSETTS

GERALD F. PURCELL
WHITMAN COLLEGE
345 BOYER
WALLA WALLA, WASHINGTON

BRUCE B. QUAYLE, SR.
MISSISSIPPI RIVER TRANS. CORP.
9900 CLAYTON ROAD
LADUE 24, MISSOURI

JOHN D. RECOB
AFIT / SSL
WPAFB BLDG. 288, AREA A
DAYTON, OHIO

JURIS REINFELDS
NASA
R-RP-N, MARSHALL SPACE FLIGHT CENTER
HUNTSVILLE, ALABAMA 35812

DARRELL G. REUM
FALCON RESEARCH AND DEVELOPMENT CO.
1441 OGDEN STREET
DENVER, COLORADO 80218

THOMAS L. RITTER
ASST. MGR. COMPUTER APPLICATIONS
COOPER BESSEMER
NORTH SANDUSKY STREET
MOUNT VERNON, OHIO

DENER A. ROE
US ARMY AIR DEFENSE BOARD
TEST SUPPORT DIVISION
FORT BLISS, TEXAS

RICHARD D. ROSS
UNI. OF MISS.
COMPUTER CENTER
UNIVERSITY, MISS.

REDMOND SAGE
WHIRLPOOL CORP.
LAUNDRY ENG. DIV.
ST. JOSEPH, MICHIGAN

GENE SANDFFUR
SCIENCE ENGINEERING ASSOCIATES
2450 MISSION STREET
SAN MARINO, CALIFORNIA

JAMES S. SASSER
HAWAII INSTITUTE OF GEOPHYSICS
COO UNIVERSITY OF HAWAII
2525 CORREA ROAD
HONOLULA, HAWAII 96822

RONALD K SAWYER
SKIDMORE, OWINGS AND MERRILL
NO. 1 BUSH STREET
SAN FRANCISCO, CALIFORNIA

JOHN SCHAFFR
STEARNS-ROGER CORP.
P.O. BOX 5888
DENVER, COLORADO 80217

BOB SCHROEDER
STEARNS-ROGER CORP.
P.O. BOX 5888
DENVER, COLORADO 80217

EDWIN W. SCHWARZ
I.B.M. CORP.
WHITE PLAINS, NEW YORK

DR. SEDAT SERDENGECTI
HARVEY MUDD COLLEGE
12TH AT COLUMBIA STREETS
CLAREMONT, CALIFORNIA

JIM SHEEHAN
STEARNS-ROGER CORP.
P.O. BOX 5888
DENVER, COLORADO 80217

JON M. SHIVE
U. S. AIR FORCE
ENT AFB COLO.
COLORADO SPRINGS, COLORADO

R. T. SIEGEL
I B M CORP.
112 E POST ROAD
WHITE PLAINS, NEW YORK

MRS. JUDITH O. SILENCE
ALLISON DIV. G.M.C.
BOX 894, PLANT 8, DEPT. 8895
INDIANAPOLIS, INDIANA 46206

BOB SMITH
STEARNS-ROGER CORP.
P.O. BOX 5888
DENVER, COLORADO 80217

GREG SMITH
I.B.M. CORP.
777 GRANT STREET
DENVER, COLORADO

NOEL SMITH
INDIANA STATE UNIVERSITY
TERRE HAUTE, INDIANA

MRS. CAROL SNAVELY
OPTICAL COATING LABORATORY, INC.
P.O. BOX 1599
SANTA ROSA, CALIFORNIA

CHARLIE C. STALLINGS, JR.
UNIVERSITY OF NEW MEXICO
RESEARCH CERTER, UNI. OF NEW MEX.
ALBUQUERQUE, NEW MEXICO

LARRY J. STARK
PUBLIC SERVICE CO. OF COLORADO
550 15TH STREET
ROOM 780
DENVER, COLORADO

FRANK J. STASTNY
MISSISSIPPI RIVER TRANS. CORP.
9900 CLAYTON ROAD
LADUE 24, MISSOURI

JOHN D. STOKES
CITIES SERVICE OIL CO.
LAKE CHARLES, LOUISIANA

FRANK C. STOLFA
CABOT CORPORATION
P.O. BOX 1101
PAMPA, TEXAS

D. SYLVESTER
NOOTER CORPORATION
1400 SOUTH 3RD ST.
ST. LOUIS, MISSOURI

RON TALLEY
FALK JORGENSEN CONSULTING ENGRS, INC.
1240 W. BAYAUD AVENUE
DENVER, COLORADO

B. G. UTLEY
I.B.M. CORP.
MONTEREY AND COTTLE ROADS
DEPT. 530
SAN JOSE, CALIFORNIA

FRANCIS J. VERLINDEN
N. C. STATE UNIVERSITY
DEPT. OF EXPERIMENTAL STATISTICS
P.O. BOX 5457
RALEIGH, NORTH CAROLINA

R. B. VLACK
I.B.M. CORP.
LOS ANGELES, CALIFORNIA

DONALD VOSS
CONSTRUCTION ESTIMATORS COMPANY, INC.
557 SOUTH SECOND WEST
SALT LAKE CITY, UTAH 84101

JERRY H. WALDON
NORTH TEXAS STATE UNIVERSITY
COMPUTER CENTER B.A. 151
DENTON, TEXAS

GARY O. WALLA
THE PROCTER AND GAMBLE CO.
MIAMI VALLEY LABORATORIES
P.O. BOX 39175
CINCINNATI, OHIO

NOEL H. WATERS
I.B.M. CORP.
777 GRANT STREET
DENVER, COLORADO

DONALD C. WEBER
KETCHUM, KONKEL, RYAN AND HASTINGS
730 KALAMATH STREET
DENVER, COLORADO 80204

BROTHER JEROME D. WEGENER
CHRISTIAN BROTHERS COLLEGE
650 EAST PARKWAY SOUTH
MEMPHIS, TENNESSEE 38104

DARRELL WILLIAMS
WICHITA STATE UNIVERSITY
WICHITA, KANSAS

DARELL D. WOLTKAMP
AERO CHART AND INFO. CENTER
2ND AND ARSENAL STREET
ST. LOUIS, MISSOURI

JAMES R. WRIGHT
TRANE CO.
LACROSSE, WISCONSIN 54601

O. G. WRIGHT JR.
PIONEER NATURAL GAS CO.
P.O.BOX 511
AMARILLO, TEXAS 79105

ROBERT S. WRIGHT
CABOT CORPORATION
P.O. BOX 1101
PAMPA, TEXAS

KAMAL F. YOUSSEF
SWINDELL - DRESSLER CO.
441 SMITHFIELD STREET
PITTSBURGH, PA.

INDUCTION LOGGING EQUATIONS COMPUTER SOLUTION

George V. Copland

Halliburton Company
Duncan, Oklahoma

At the outset, I would like to say that this paper to be presented does not concern itself so much with the answer solution to the induction logging response equation as it does with the organization of the problem itself. The problem is not new and, no doubt, has been set up and solved on numerous occasions, not only by ourselves but by others in the same field of study. About fourteen years ago, our Company rented a 604 from IBM and some preliminary work was done. Like airplanes, the greater the model number, the better, faster and higher the computer, so soon we were obliged to try a 650. The problem seemed to be more of a card handling project than a computer solution to equations. From the results of many months of computing, some studies were made and a logging tool was designed and introduced in the field. This tool, in modified form, is essentially the type used today.

In the last several years the induction logging tool has been more widely used and accepted. The Engineering group now had available a 1620, 20K machine with a 1622 card read punch. This computer was used to optimize the design of our logging tools. Many various multi-coil systems were investigated in this optimization program.

Halliburton Company is primarily a service company to the petroleum industry. We perform various services the customer may require. These include cementing, fracturing, chemical services as well as electrical logging of the formations. These may be either DC potential contact logs, radioactive logs or induction type logs using AC currents. This paper is a discussion of the computer programming required to design induction logging tools. The next series of slides show the generalized form of the equation involved.

For integration of the curve, let us examine the left position of the curve showing this stepwise incrementation. As you can see, the error is always positive, so the solution is to just reduce the increment and improve accuracy. Well, it helped, but not much! So, I tried something different and I'm sure it's not original, but if you calculate every other step and multiply by two, as shown in the right side of the graph, it works nicely. Since we knew the answer and the accuracy desired, some idea was known as to how far out to infinity to go before the change in the incremented value became insignificant. This turned out to be from five to seven minutes per value timewise.

Upon examination of the Response Equation, one finds the main time consuming portion of the problem will be in the integration of the R values and Z values. Since we were still operating with the 1620 BC 20K, it was necessary to compute these tables and store them in card form and use manual look-up. First of all, we needed to know what steps to do the integration and how far we needed to carry out the calculations since we were going to infinity. I don't know any other way to do this but to start running and

and see how well it works. I would like to recognize the mathematicians present who have already figured out thst one can integrate the equation directly without resorting to long drawn-out incrementation techniques. This did give ma check and is quite direct, however, it didn't work - for me at least - when I tried to integrate the Z equation. So, I still had the problem of generating tables and a quick way of checking myself as I went along. It is somewhat discouraging to know the answer to the R equation is 1.0 and take so long to prove it on a 1620.

Another problem was how to set the program up so anyone could break into the problem and use the computer, since most work was to be done after regular hours.

The program was arranged so the output card carried all the information required by the problem and could be used to input data for the next value. If, say at night, when someone wanted to use the computer, all one had to do was stop the computation and clear the punch. The last output card was added to the program deck and when one wanted to restart the problem, all that was necessary was to reload this deck. This required reloading the program but since it was a small deck, it required only a few minutes. In case of card jams at night, the last card in the punch output hopper could be used to restart the calculation with nothing lost.

I found that the first card which I called the zero card was easily handled by a shorter program and therefore a separate program was used to calculate the first card which in turn started the second continued program. Obviously, I did not wish to integrate any further than required in either the R or Z direction. The answers were periodically plotted to obtain the graph shown. You will note at approximately R = 12, the curve has begun to flatten.

Most of you are probably aware of the curve fit program in the 1620 Users Group Library. This program was used to check the plot and it was found a curve fit could have had values of from 4 to 12 and 10,000. I assumed, for all practical purposes, the answer was 1.0 at 10,000. This worked out rather nicely and a hyperbolic curve was determined that gave us quite good correlation in the area, especially from 8 to 12. Some values were calculated and by slight adjustment of the resulting constants in the hyperbolic equation, I could match the curve to the accuracy required by the tables. These constants were also used in the problem solution, as I will discuss a little later. The R tables were calculated stepwise integration from 0. to 12. and by a hyperbolic equation from 12. to 60. A set of cards was assembled covering this range and listed on a 407 for tabular look-up.

The Z table calculations were handled in a similar manner as the R tables. As it turned out, it was easier to calculate the zero card separately. By using this card as a starting data for a program, I started calculating Z tables in a likewise fashion. When the computer got to Z = 1.0 it went into the error routine. After checking the problem, program and equation, I found that the equation was discontinuous at Z = 1.0. Since the computer was unable to calculate this value, I simply told it what the value of R = 0.0 and Z = 1.0 was, and continued on. Since I was unable to integrate the equation in respect to Z directly and after plotting some of the values as the calculation progressed, I decided that the answer to the problem was .50 at infinity.

I decided to make a 3-dimensional graph of the vertical response equation using swab sticks cut to lengths and a piece of 1/2" black plastic sheet for a base. The equation is quickly solved when you are solving for points directly. I wrote a program that calculated these points and found that about 20 minutes running time was all that was required to give us the model shown. I included the 1/2" dimension for thickness of the board and the answer so that the sticks could be cut directly. An interesting point here was a short program was written for our automatic drill press in the main production facility to drill the base. In this program, it turns out it is faster on this particular machine to drill every fourth hole - back up to the beginning, increment a quarter unit and drill another set of holes 4 units apart.

By now I had decided our accuracy adequate but I desired to reduce the time required for calculations. By examining the three dimensional form we obtained from the plot, I was able to estimate to some degree a better technique to speed up the incrementation. If you will note the bulk of the graph weight lies in the volume from R = 4.0 and Z = 2.0. One might be able to speed up the calculations by increasing the size of the incrementation after exceeding these limits. This was done and we saved about 25% in computer time over the R table calculations in this manner. I am sorry we had not made an R table three-dimensional graph to begin with. As in the R tables, we used the curve fit program and obtained different values for the constants in the hyperbolic equation. These were adjusted slightly and values were calculated out to Z = 60.0.

So now I have a deck of cards for R values from 0.0 to 60.0 and Z values from 0.0 to 60.0. I also listed these so they could be used in tabular look-up.

We decided to check our previous work done on induction logging tools. I wrote a program which used the typewriter to tell me which cards were needed to be looked-up from data storage. This amounted to values for GR and 12 values for GB. We would load these cards into the card read punch and read them into the program. These cards were checked and interpolated to give a little closer reference values. These solutions check nicely with our previous work and actually showed some inaccuracies in the previous work. It turned out that it was rather tedious searching for cards, loading them in the card read punch, processing and inserting them back into the card boxes. The computer could calculate the problem faster than we could handle the card look-up system.

By this time our 20K computer had become overloaded from two sources. The time required for compiling and loading programs from cards and also our problems were simply requiring more than 20K storage Fortranwise to solve. Some of our larger programs were written in SPS and even these exceeded our core capacity. Rather than get additional 20K on our existing computer, we decided to get a 1620, 40K with disk drive. Along with this equipment came the Clock, Monitor and Fortran IID.

The first problem facing us in our induction log problem was to put our R and Z tables on disk, so we could retrieve it under program control. As you know, the Fortran IID is a formated input program that will not accept the free form output that we had from our Forcom routine. We therefore were faced with taking our original free form output cards and converting them to a format suitable for our computer. The AFIT program would allow us to use free form input and E FORMAT output. By putting a record mark in 20,000, we

used this program to convert the decks, consisting of cards from 0. to 60. for both R and Z values in steps of .01. This gave us a total of 12,000 cards which would require 60 cylinders to store in the form used by the Record, Fetch statements of IID. In storing data on a disk you need to store the data as a block in some section and move it down into working storage when you run the problem. This work would have required 120 cylinders which is in excess of the 100 cylinders available on a disk. Looking at the method, we obtained the values of from 12. to 60. by hyperbolic curve fit. I decided to calculate the higher values directly instead of table look-up. When I ran the multi-coil problem I found values as high as 45 not uncommon but we rarely went over 60.

The DFINE statement is the heart of the working storage assignment. You will recall that in the Manual, it says you can reserve up to a maximum of 99 work cylinders. I decided to use 32 work cylinders for working area. I had defined the disk sector address of the work cylinders as sector zero. When I tried to load this control card, the computer gave an error 13 message, "insufficient available storage for specified work cylinder". You will remember the DIM table starts in Cylinder 24. I decided the work cylinder area could not be greater than 24. If it was greater, then it would require it to jump over the DIM and equivalence table. Whether this is true of not, I did not determine. I decided to use 20 working cylinders for data storage and this worked nicely. A short program was written using the Record statement which assigns the data card in sequence in the working cylinder area and assigned a record number to this data. By defining the work length 10 or less, it put a card per sector per record number. Also, we were told by IBM that record number 1 starts in sector 219. I have yet to find this statement written in the Manual as such. 219 from 3999 leaves us 3780 sectors for data storage and this is the number of cards we used to load into data storage. If you note in the 1311 Manual, the time required to execute a Write Disk statement is two seconds. Since each card is handled individually, the 3780 cards took a little over two hours to load two boxes of data. Dollarwise, this takes $70.00 to load two boxes of cards onto disk. Kind of expensive when you think of it this way.

After loading the data cards into working cylinder areas on the disk, I wished to move them to permanent storage area on the disk so the program could recall them when running the problem. This is required because the Fortran compiler compiles in the first working cylinders and destroys the first few cylinders of tables. I also found that a couple of other working cylinders were used during this compilation. I will get to this in a minute. You have two choices to move data from working cylinders up to permanent storage under Monitor control. One is by DLOAD and the other is by DCOPY. I used the DLOAD first to transfer this block of data up into permanent storage. I chose to move 3780 sectors by differences in sector address. The routine moved 780 sectors so I assumed that a maximum of 999 sectors could be moved by DLOAD control card. In using the DCOPY control card, you can successfully move 3780 sectors and file protect. However, when you type out the Availability list you will note that it shows this area to be open and usable even though it has been file protected and has data stored therein. I was careful to avoid this area, so as not to destroy the tables and have to reload.

I had to rewrite the multi-coil program on the induction logging response equation in Fortran IID, I decided to make this a generalized program which would encompass 4, 5, 6 and 8-coil designs. Since I did not want to get involved with the complexities of subscript notation, I defined all of my variables directly, that is, R1T1, R1T2, etc. After compiling this program, it was

found the program required in excess of 50,000 core positions. As shown by the block diagram, the program was split up by making the 8-coil portion of the GB section a subprogram on call as this would be used the least. The data R and Z tables were split up in two smaller groups of from 0 to 18.99 for R tables and 0 to 18.80 for Z tables. If the problem required values look-up greater then either of those stored in the R and Z tables, it used the hyperbolic equation constants previously derived to calculate these values. This master program was stored on Disk under the name MULTIC.

The procedure to run the problem is to do a *DCOPY and move the data from permanent storage to working cylinders. Next, do a XEQS MULTIC which calls the program off of disk and moves it into core for execution. The input data cards are loaded as required by the program. A 1620 Users Group program is available called GETREC which will do this DCOPY under Fortran Program control. At the time I wrote the MULTIC program, the GETREC program was not available.

Several things you cannot do are these: You cannot do a *DCOPY after XEQS program because the DUP routine destroys the program. You cannot do a FORX because in compilation, this evidently destroys an area of Cylinder 3 and Cylinder 10. I discovered this after I couldn't use the Z tables which were stored above Cylinder 9. If you watch the index wheel in the 1311 when compiling, you will notice it goes to Cylinder 10 at least once. Sometimes the input data would be out of range and we wished to restart the program by a branch command on the typewriter. The program communication area starts in position 02218. The address beginning at 02222 is the restart address of the program. A branch using this address or 490226 will start the program if the check stop light is not on.

Information available in this area starting with 02218 is Floating Point Word Length, Fixed Point Word Length, Starting Address of Main Line Program, Starting Address of Common Area, Number of Words in Logical Area, Number of Logical Records, Word Length and Record Length.

The running time of this program for a 6-coil design was about 4 minutes. In generation of R tables, it took approximately 250 hours computer time and about 200 hours for the Z tables. A good average for the values that had to be looked-up would have required about 100 hours of integration time per value. For an 8-coil design problem you would have had 16 R values and 32 Z values to look-up. This would have been 48 values if integrated directly at 100 hours per value, which gives us about 4800 hours for value determination. Since it only takes about 4 minutes to run this problem by using table look-up, we effectively have a time savings of about 72,000 to one. If I had had a 7090 or equivalent type computer, I might have gone ahead and done the proglem the long way, not aware of the time savings available by table look-up technique. We not only have a solution which is quite useable to make configuration studies, but we also have a set of tables which can be used without the computer by an engineer to solve this problem.

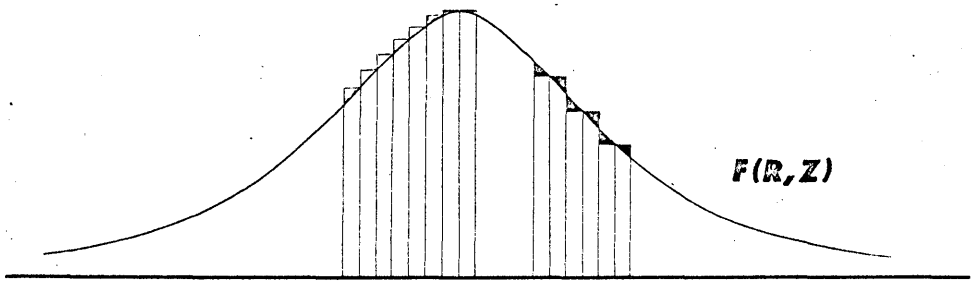
Our computer time runs in the order of $40.00/hr. and at approximately 4800 hours solution the long way is some $200,000. One time through (using the Disk Storage) costs about $3.00.

We have run many hundreds of solutions to optimize this design, so you see the justification of this approach.

RADIAL RESPONSE CURVE - R VALUES

$$\sum R = \sum_{R=0.0}^{R=12.0} \sum_{Z=-\infty}^{Z=+\infty} F(R,Z)\,\Delta R\,\Delta Z$$

WHERE: $$F(R,Z) = \frac{R^3}{\left[R^2+(1+Z)^2\right]^{3/2}\left[R^2+(1-Z)^2\right]^{3/2}}$$



Incremention Steps for Integration of Curve

THE DIRECT AND CYCLIC JACOBI METHODS WITH FADEEVA's CORRECTION ALGORITHM FOR A REAL SYMMETRIC MATRIX

HENRY E. FETTIS
JAMES C. CASLIN

APPLIED MATHEMATICS RESEARCH LABORATORY

JULY 1966

Project 7071

AEROSPACE RESEARCH LABORATORIES
OFFICE OF AEROSPACE RESEARCH
UNITED STATES AIR FORCE
WRIGHT-PATTERSON AIR FORCE BASE, OHIO

THE DIRECT AND CYCLIC JACOBI METHODS WITH FADEEVA's CORRECTION ALGORITHM FOR A REAL SYMMETRIC MATRIX

One way of formulating the eigenvalue-eigenvector problem for a square matrix A is to find a matrix T such that

$$T^{-1} AT = \Lambda \tag{1}$$

where Λ is either a diagonal or "Jordan-Normal" matrix. If Λ is diagonal* the diagonal elements are the eigenvalues and the columns of T the eigenvectors of A. If A is real-symmetric, then T is orthogonal which implies $T^{-1} = \tilde{T}$. In this case the existence of the diagonal form is assured, even for multiple eigenvalues.

The Jacobi method finds the matrix T by an iterative process. The diagonal form is realized by performing successive rotations on 2x2 submatrices of A in such a way as to eventually anihalate all the off-diagonal elements (to within a specified tolerance). Two schemes for doing this are explained later. To understand the basic transformation the corresponding problem for a 2x2 matrix is considered. Let

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{bmatrix}; \quad x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}. \tag{2}$$

Consider now the quadratic form associated with A, namely

$$Q(x_1, x_2) \equiv \tilde{x}Ax = a_{11} x_1^2 + 2a_{12} x_1 x_2 + a_{22} x_2^2 \tag{3}$$

The curves $Q(x_1, x_2) = \text{const}$ represent conic sections in the (x_1, x_2) plane. From analytic geometry, we know that there exists a rotation of axes such that, in the new system, the $x_1 x_2$ term is absent. Hence, in the rotated system the associated matrix must be in the form

* For arbitrary matrices, the diagonal form is not always possible.

$$\begin{pmatrix} \bar{a}_{11} & 0 \\ 0 & \bar{a}_{22} \end{pmatrix} \tag{4}$$

The transformation which rotates the axes through an angle θ is, evidently, given by

$$x = O\,\bar{x} \tag{5}$$

where

$$O = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \tag{6}$$

and the new quadratic form by

$$\bar{Q}(\bar{x}_1, \bar{x}_2) = \tilde{x}A\bar{x} = x(\tilde{O}AO)x \tag{7}$$

Hence the matrix

$$\bar{A} = \tilde{O}AO \tag{8}$$

must be diagonal. The expression for θ is well known:

$$\tan 2\theta = \frac{2a_{12}}{a_{11} - a_{22}} \,. \tag{9}$$

The above principle constitutes the basis of the classical Jacobi method and its modifications. In all versions, the given matrix A is transformed to a new matrix A_1 defined by

$$A_1 = \tilde{T}AT \tag{10}$$

The matrix T is made up of a 2x2 submatrix of the form O, with one's on the remaining diagonal position - and zero's elsewhere, the elements of O being so chosen that some pair of corresponding off-diagonal elements of A_1 will vanish. The matrix A_1 is again transformed in such a way as to create zeros in another pair of off-diagonal positions. In so doing, the elements,

originally zeroed by the first transformation will again attain non-zero values. However, it can be proven that, by repeated applications of such transformations, all of the off-diagonal elements can be reduced to arbitrarily small numbers. When this point has been reached, the eigenvalues can be read from the diagonal elements, and the eigenvectors found by combining the successive transformations matrices into a single one.

Two versions of the above mentioned have been programmed for the 1620. The first, based on the classical approach suggested by Jacobi, always anihilates the off-diagonal element of greatest magnitude. The second version, sometimes called the cyclic method "anihalates" the off-diagonal elements in some systematic order, for example:

$$\begin{matrix} a_{12} & a_{13} & \cdots & a_{1n} \\ & a_{23} & \cdots & a_{2n} \\ & & \cdots\cdots & \\ & & & a_{n-1,n} \end{matrix} \qquad (11)$$

skipping any which are less than a specified tolerance, and making as many passes as are necessary to reduce all to the required smallness.

In general, no appreciable difference in time between the two versions could be detected, although in some cases the cyclic method was somewhat faster.

Several refinements are included in the present programs. Equation (9) was replaced by the following two

$$\cos\theta = \sqrt{\tfrac{1}{2}\left[1 + \frac{\alpha}{\sqrt{\alpha^2+\beta^2}}\right]} \qquad (12)$$

$$\sin\theta = \frac{|\beta|}{\beta}\sqrt{\tfrac{1}{2}\left[1 - \frac{\alpha}{\sqrt{\alpha^2+\beta^2}}\right]}$$

where

$$\alpha = (a_{ii} - a_{jj})/2 \qquad (13)$$

$$\beta = a_{ij}$$

and (i,j) are the indeces of the particular elements to be anihalated. These were only used as long as β was large enough that no loss of accuracy resulted from the subtraction under the radical. Thereafter $\cos\theta$ and $\sin\theta$ were computed from the following series

$$\cos\theta \tilde{=} 1 - \tfrac{1}{2}\left(\frac{\beta}{2\alpha}\right)^2 + \frac{11}{8}\left(\frac{\beta}{2\alpha}\right)^4 + \cdots + \tag{14}$$

$$\sin\theta \tilde{=} \frac{\beta}{2\alpha}\left[1 - \frac{3}{2}\left(\frac{\beta}{2\alpha}\right)^2 + \frac{31}{8}\left(\frac{\beta}{2\alpha}\right)^3 + \cdots + \right.$$

Finally a device due to Fadeeva([1], p 484) was added which improves the final approximations to the eigenvalues and eigenvectors to within a higher tolerance than that attained by the method itself. The correction takes the form

$$\lambda_i \sim a_{ij} \sum_{\substack{j=1 \\ j\neq i}}^{n} \frac{a_{ij}^2}{a_{ii} - a_{ji}} \tag{15}$$

with a similar correction to the components of the computed eigenvectors. The correction requires modification if there are multiple eigenvalues. Also, the algorithm is only valid if the off-diagonal elements are small compared to the lowest eigenvalue. When dealing with matrices such as the Hilbert matrix in which there is wide separation in the greatest least eigenvalue, a knowledge of the magnitude of the lowest one is helpful.

The method has been tested on segments of the Hilbert matrix. A check on the accuracy of the results can be made by comparing the product of the eigenvalues with the value of the determinant, for which the exact expression is known [2]. A second test matrix due to W. Frank [3] was tried. In this example the exact eigenvalues were known which allowed a direct comparison to be made between the uncorrected and corrected values. In each case the results proved that the corrections were of the order of magnitude of the original tolerance squared.

//

REFERENCES

[1] Faddeev, D.K. and Faddeeva, V.N. "Computational Methods of Linear Algebra", W.H. Freeman & Co., San Francisco (1963).

[2] "Contributions to the Solution of Systems of Linear Equations and the Determination of Eigenvalues", p 106, A.M.S. 39, N.B.S. (1954).

[3] Frank, Werner L. "Computing Eigenvalues of Complex Matrices by Determinant Evaluation and by Methods of Danilewski and Wielandt", SIAM Journal, v 6 #4 (December 1958).

EXHIBIT MAIN

```
C       MAIN PROG TO TEST VARIABLE PRECISION JCBI SUB
        DIMENSION A(24,12),B(12,12),EE(12),Q(12)
        COMMON A,B,EE
1       ACCEPT 102,ER
        ACCEPT 101,N
        N1=N+1
        NN=N+N
        IF(SENSE SWITCH 1)200,203
200     DO 201 I=1,N
        DO 201 J=1,I
        READ 102,A(I,J)
201     A(J,I)=A(I,J)
        GO TO 202
203     CALL LI(DET,N)
        CALL MATX(  N)
202     CALL JCBI(A,N,ER)
205     CALL IMPEV(N,EE,Q)
        S2=1.0
        S1=1.0
        DO 206 K=1,N
        S1=S1*EE(K)
206     S2=S2*A(K,K)
        S3=S1-DET
        S4=S2-DET
        PUNCH 120
        PUNCH 121,DET,S2,S1
        PUNCH 122, S4,S3
        CALL VEC(A,B,N)
        PUNCH 112
        PUNCH 115,(A(I,I),Q(I),EE(I), I=1,N)
        PUNCH 109
        DO 26 J=1,N
        DO 25 I=N1,NN
        K=I-N1+1
25      PUNCH 110,A(I,J),B(K,J)
26      PUNCH 106
        IF(SENSE SWITCH2)6,10
6       PUNCH 114
114     FORMAT(15X25HINPUT MATRIX DIAGONALIZED/)
        DO 7 I=1,N
        DO 8 J=1,N
8       PUNCH 113,A(I,J)
7       PUNCH 106
10      PAUSE
        IF(SENSE SWITCH 3)9,1
9       DO 40 I=N1,NN
        DO 40 J=1,N
        K=I-N1+1
        IF(I-J-N)41,42,41
42      A(K,K)=EE(K)
        GO TO 40
41      A(I,J)=B(K,J)
40      CONTINUE
        GO TO 205
```

```
113   FORMAT(E26.19)
106   FORMAT(/)
112   FORMAT(4X17HAPPROX-EIGENVALUE14X10HCORRECTION15X14HNEW-EIGENVAL
109   FORMAT(/15X7HVECTORS/)
102   FORMAT(E14.7)
101   FORMAT(I2)
104   FORMAT(/15X17HEXACT EIGENVALUES/)
110   FORMAT(2E30.19)
115   FORMAT(E26.19,1XE26.19,1XE26.19)
120   FORMAT(3X17HEXACT DETERMINANT10X12HBEFORE CORR.15X11HAFTER CORR.)
121   FORMAT(E26.19,2(1XE26.19))
122   FORMAT(26X,2(1XE26.19))
      END
```

EXHIBIT A

```
C       DIRECT JACOBI METHOD
C       LARGEST OFF DIAGONAL ELEMENT METHOD
C       A=A(I,J) , A REAL SYMMETRIC MATRIX DIMENSIONED (2N,N)
C       WHERE N= ORDER OF MATRIX, ER=TOLL.
C       AFTER CONVERGENCE THE DIAGONAL OF THE UPPER NXN MATRIX
C       WILL CONTAIN THE APPROXIMATE EIGENVALUES OF A(I,J)
C       THE COLUMN OF THE LOWER HALF WILL CONTAIN THE APPROXIMATE EIGENVECTORS.
        SUBROUTINE JCBI(A,N,ER)
        DIMENSION A(24,12)
        N1=N+1
        N2=N-1
        NN=N+N
   27   DO 7 I=N1,NN
        DO 7 J=1,N
        IF(I-J-N)6,5,6
5       A(I,J)=1.0
        GO TO 7
6       A(I,J)=0.0
    7   CONTINUE
   20   B=A(2,1)
        M=2
        L=1
        IF(N-2)26,26,25
   25   DO 9 I=3,N
        I1=I-1
        DO 9 J=1,I1
        IF(A(I,J)*A(I,J)-B*B)9,9,8
    8   B=A(I,J)
        M=I
        L=J
    9   CONTINUE
   26   IF(B*B-ER*ER)24,24,21
   21   AL=(A(L,L)-A(M,M))/2.0
        IF(AL)28,22,28
   28   X=B/(2.0*AL)
        Y=X*X
        IF(Y-.0001)23,23,22
   23   C=1.0-.5*Y+1.375*Y*Y-4.3125*Y*Y*Y
        S=X*(1.-1.5*Y+3.875*Y*Y-11.6875*Y*Y*Y)
        GO TO 12
   22   R2=AL*AL+B*B
        R=SQRTF(R2)
        C2=.5*(1.0+AL/R)
        S2=1.0-C2
        C=SQRTF(C2)
        IF(B)10,11,11
   10   S=-SQRTF(S2)
        GO TO 12
   11   S=SQRTF(S2)
   12   DO 13 I=1,NN
        D=C*A(I,L)+S*A(I,M)
        E=-S*A(I,L)+C*A(I,M)
        A(I,L)=D
13      A(I,M)=E
   14   DO 15 J=1,N
        D=C*A(L,J)+S*A(M,J)
        E=-S*A(L,J)+C*A(M,J)
        A(L,J)=D
   15   A(M,J)=E
40      IF(N-2)24,24,20
   24   RETURN
```

```
C       CYCLIC JACOBI METHOD
C       A=A(I,J) , A REAL SYMMETRIC MATRIX DIMENSIONED (2N,N)
C       WHERE N= ORDER OF MATRIX, ER=TOLL.
C       AFTER CONVERGENCE THE DIAGONAL OF THE UPPER NXN MATRIX
C       WILL CONTAIN THE APPROXIMATE EIGENVALUES OF A(I,J)
C       THE COLUMN OF THE LOWER HALF WILL CONTAIN THE APPROXIMATE EIGENVECT

        SUBROUTINE JCBI(A,N,ER)
        DIMENSION A(24,12)
        N1=N+1
        N2=N-1
        NN=N+N
        NT=(N*N2)/2
        DO 7 I=N1,NN
        DO 7 J=1,N
        IF(I-J-N)6,5,6
5       A(I,J)=1.0
        GO TO 7
6       A(I,J)=0.0
7       CONTINUE
19      K=0
        DO 17 L=1,N2
        L1=L+1
        DO 17 M=L1,N
        B=A(L,M)
        IF(B*B-ER*ER)16,16,21
21      AL=(A(L,L)-A(M,M))/2.0
        IF(AL)28,26,28
26      C2=.5
        GO TO 25
   28   X=B/(2.0*AL)
        Y=X*X
        IF(Y-.0001)23,23,22
23      C=1.0-.5*Y+1.375*Y*Y-4.3125*Y*Y*Y
        S=X*(1.-1.5*Y+3.875*Y*Y-11.6875*Y*Y*Y)
        GO TO 12
   22   R2=AL*AL+B*B
        R=SQRTF(R2)
        C2=.5*(1.0+(AL/R))
   25   S2=1.-C2
        C=SQRTF(C2)
        IF(B)10,11,11
   10   S=-SQRTF(S2)
        GO TO 12
   11   S=SQRTF(S2)
   12   DO 13 I=1,NN
        D=C*A(I,L)+S*A(I,M)
        E=-S*A(I,L)+C*A(I,M)
        A(I,L)=D
   13   A(I,M)=E
   14   DO 15 J=1,N
        D=C*A(L,J)+S*A(M,J)
        E=-S*A(L,J)+C*A(M,J)
        A(L,J)=D
15      A(M,J)=E
        IF(N-2)24,24,17
   16   K=K+1
   17   CONTINUE
        IF(K-NT)19,24,24
   24   RETURN
        END
```

EXHIBIT C

```
C     FADEEVA,S EIGENVALUE CORRECTION ALGORITHM

      SUBROUTINE IMPEV(N,EV,RESUD)
      DIMENSION A(24,12),EV(12),RESUD(12)
      COMMON A
      DO 6 I=1,N
      W=0.0
      DO 5 J=1,N
      IF(I-J)7,5,7
7     Y=A(I,I)-A(J,J)
      X=A(I,J)
      Z=X*X
      IF(Y)3,4,3
4     PRINT 100
      GO TO 8
3     W=W+Z/Y
5     CONTINUE
      RESUD(I)=W
6     EV(I)=A(I,I)+W
8     RETURN
100   FORMAT(56HMATRIX HAS MULTIPLE EIGENVALUES-CORRECTION METHOD FAILS.
     1/)
      END
```

EXHIBIT D

```
C       FADEEVA,S EIGENVECTOR CORRECTION ALGORITHM

        SUBROUTINE VEC(A,B,N)
        DIMENSION A(24,12),B(12,12),EV(12)
        COMMON A,B,EV
        NN=N+N
        N1=N+1
        DO 6 K=1,N
        DO 4 M=1,N
        IF(K-M)2,3,2
3       EV(M)=1.0
        GO TO 4
2       EV(M)=A(M,K)/(A(K,K)-A(M,M))
4       CONTINUE
        LL=0
        DO 6 I=N1,NN
        LL=LL+1
        S=0.0
        DO 5 J=1,N
5       S=S+A(I,J)*EV(J)
6       B(LL,K)=S
        RETURN
        END
```

EXHIBIT E

```
C       COMPUTES THE DETERMINANT OF A HILBERT MATRIX (EXACT)

        SUBROUTINE LI(DET,N)
        G=1.0
        FAC=1.0
        C=N
        CN=C**N
        C1=C**2
        N1=N-1
        DO 2 K=1,N1
        FK=K
        FAC=FAC*FK
        J=N-K
        G=G*((FAC**2)/(C1-FK**2)**J)
2       CONTINUE
        DET=G/CN
        RETURN
        END
```

EXHIBIT F

```
C       COMPUTES ELEMENTS OF HILBERT MATRIX

        SUBROUTINE MATX(  N)
        DIMENSION A(24,12)
        COMMON A
        DO 2 I=1,N
        X=I
        DO 2 J=1,I
        Y=J
        A(I,J)=1./(X+Y-1.0)
2       A(J,I)=A(I,J)
        RETURN
        END
```

TRANSMISSION LINE SAGS AND TESNIONS

Perhaps the first person faced with the problem of tension and corresponding sags in a line stretched between two points may have been named ADAM. It is conceivable that Eve, having gotten Adam to sample the apple, caused an awareness of the need for fig leaves, and a consequent need for a fig line on which to hang the newly washed apparel. A grape vine being in sight, spanning some distance between two trees, may have offered the best solution Adam could master under the circumstances. Most likely, being a male, he would rather see the fig leaves on the wash line than on Eve - but be that as it may - a line strung between two rigid supports has both sag and tension and because of these characteristics, Adam may have had some trouble. Perhaps one day Eve hung one too many wet fig leaves on the grape vine and lo and behold several of the freshly laundred fig leaves began to drag in the dirt, making Eve very unhappy indeed. Ole Adam, seeing the predicament, figured he would fix that. So he quietly slipped around to one end of the grape vine and hitched it up a notch. Whoosh! The vine parted and Eve's wash really in the dirt - you can imagine what Adam got for that!

Ever since this fictitious event, we have been worrying about sags and corresponding tensions and vice versa. Through the intervening years, mathematicians and physicists have formulated the laws governing lines strung between supports. This turned out to be what is called a catenary and it is fairly simple to calculate the tension for a certain sag, or, if necessary, the sag for a given tension. Since temperature caused the line to expand as it got hotter and contract when it got colder, the sag would not stay put and neither would the tension. About this time some one found out that the line stretches some when you pull on it and if you pull too hard, it won't return to its original length. And, of course,

if it is pulled harder still, it breaks, and you start over. Mathematicians and physicists are very smart people. They found a way to get the temperature and stretch into a formula for determining the sag and tension when the temperature and the stretch characteristics of the line are given.

The line crews being people, and not caring to string a line in freezing weather with 1/2-inch of ice and with an 8-mile per hour wind blowing on it just to provide adequate ground clearance for a certain hi-voltage under these conditions, asked them (the physicists and mathematicians) to go back and figure some more. They preferred to string this wire on a nice sunny, 70° day at some tension that would guarantee that when that same wire was subjected to zero-degree weather, and had a certain amount of ice formed on it, the tension would not exceed a given value. Also, that the sag could be depended upon to still clear the ground by the desired amount.

This caused the physicists and mathematicians no end of trouble. About the only way they could do this was to start with the specified tension at the loaded condition, determine the permanent stretch, the elastic stretch, the long time creep and then the corresponding sag. Working backward from the loaded condition to the condition of a nice sunny day to determine the tension to use, became a cut-and-try problem until the conditions of stringing on the warm sunny day would produce the desired result on the worst anticipated day. The cut-and-try method involved a lot of laborious calculations and good guessing to minimize the number of trial calculations.

Many methods have been devised to simplify this chore. One of the methods was devised by Mr. Francis J. Hubert, of the Division of Water and Power, City of

Los Angeles. The method of computation used in this computer program is based on his technical paper, "Simplified Sag-Tension Equations for Power Lines." The equations developed by Mr. Hubert are empirical equations that give results within 1% accuracy and are easily adapted to computer programming. See EXHIBIT 1.

The equations used are based on the diagram shown here. From this diagram, it can be seen that L is the span length, the right-hand support is elevation-high, the support on the left elevation-low. Tension will be that in the conductor at the high support. The solution gives this tension, the vertical sag below the low support, and the distance from the low support to the lowest point of the sag. Of course, for supports of equal elevation, the low point is mid-point of the span.

It was desirable in developing this program to have a direct and simple input format, a complete and readable output, covering the normal range of expected temperatures, and to provide automatic conformance to the National Electrical Safety Code Rules. Rule 1 of the safety code stipulates that the maximum tension shall not be more than 60% of ultimate strength. Rule 2, the tension at 60°F, without external load, shall not exceed 25% of ultimate tension in the final unloaded condition. Rule 3, the tension at 60°F, without external load shall not exceed 35% of ultimate tension in the initial unloaded condition.

Most utilities are somewhat more conservative and, to provide more safety from Aeolian vibration, apply the last two rules, not at 60°F, but at the lowest temperature expected (usually zero degrees).

The Program Flow Diagram, Exhibit 2, shows the method used to insure conformance to the safety rules previously mentioned. TMAX is an input quantity based on Rule 1. If TMAX is entered with a value larger than R1 percent of ultimate, the

program automatically reduces it to this value. A calculation is then made to find the corresponding tension for the unloaded final condition at the same temperature. This value is compared with R2 percent of ultimate. If the calculated tension is less than R2% of ultimate, the program proceeds. If the calculated tension exceeds R2 percent of ultimate, TMAX is reduced and recalculation takes place until Rule 2 is satisfied.

Rule 3 percent of ultimate is then used to calculate the stringing tension at the same low temperature. This tension is then compared with the previously used TMAX. If the calculated value of tension T4 is less than that produced by TMAX, the program automatically makes TMAX = Rule 3 percent of ultimate with the assurance that if TMAX passed Rule 1 and Rule 2, and R3 times ultimate was the largest value that would pass Rule 3, the new value of TMAX would consequently pass all three rules. If T4 were greater than that produced by TMAX, then Rule 3 would automatically be passed. With the rules having been cleared with an acceptable TMAX, the calculations proceed through the specified number of temperature increments to produce the output results by following the Program Flow Diagram, Exhibit 3.

Data input is accomplished by 5 input cards--3 for identification, 1 for conductor data, and 1 for span data. A form was developed for punching data as illustrated in Exhibit 4.

This exhibit is a picture of 5 input cards with the data arrangement shown. Note that the first 3 cards are identification. Any wording punched will be reproduced on the output. Cards 4 and 5 are conductor and span data cards. They include:

Conductor data	Span data
Type	No.
Size	Length
Strand	Elevation-Hi
Diameter	Elevation-Lo
Area of Conductor	Wind pressure
Area of steel	Thickness of Ice
Area of aluminum	Initial temperature
Weight	Temperature increment
Ultimate tension	Final temperature
Maximum tension allowed	Rule 1 %
Creep constant	Rule 2 %
Largest Span No.	Rule 3 %
	Loading
	Loading Code

The program is stored on disk pack and when called into the computer, excecutes the program with the above data to produce the output as follows:

Alternate No. 1 EXHIBIT 5

- only final sags and tensions calculated

Alternate No. 2 EXHIBIT 6

- without including long term creep

Alternate No. 3 EXHIBIT 7

- Enter exact values of modulus of elasticity and coefficient of expansion - otherwise it is calculated internally.

Switch 4 is not used in the program.

Alternate No. 4 EXHIBIT 8

With all switches off:

- Full output, final and stringing tensions and sags, with 10-year creep, using calculated values of modulus of elasticity and coefficient of expansion.

Calculation of each span with full output takes about 3½ minutes with the 1443 on-line printer.

Accuracy has been proven by comparison with other calculations. One notable comparison was on a proposed long span of 10,000'.

The results compare very closely with the results obtained from the Copperweld Company for the same span and conditions. EXHIBIT NO. 9.

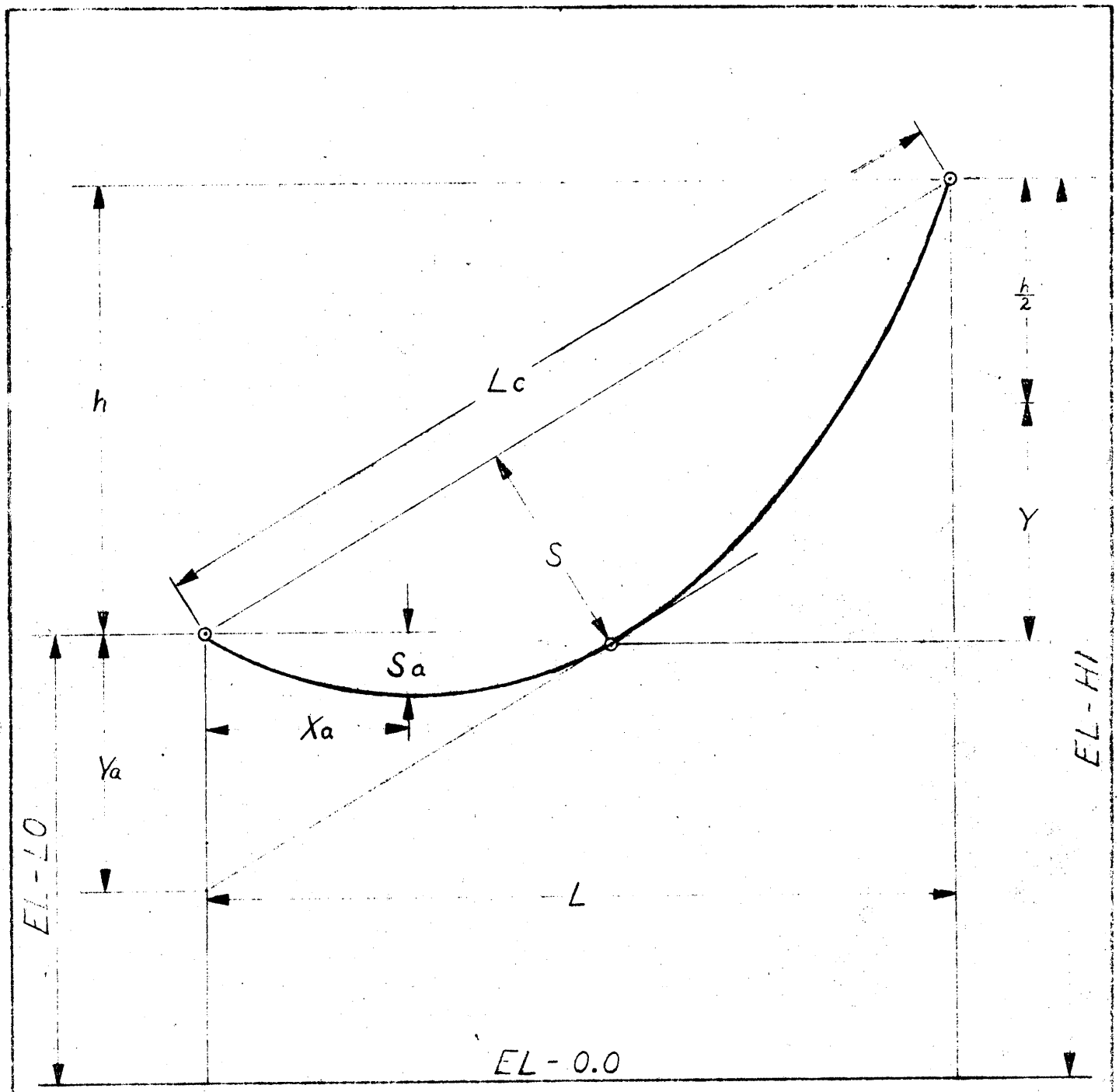


EXHIBIT 1

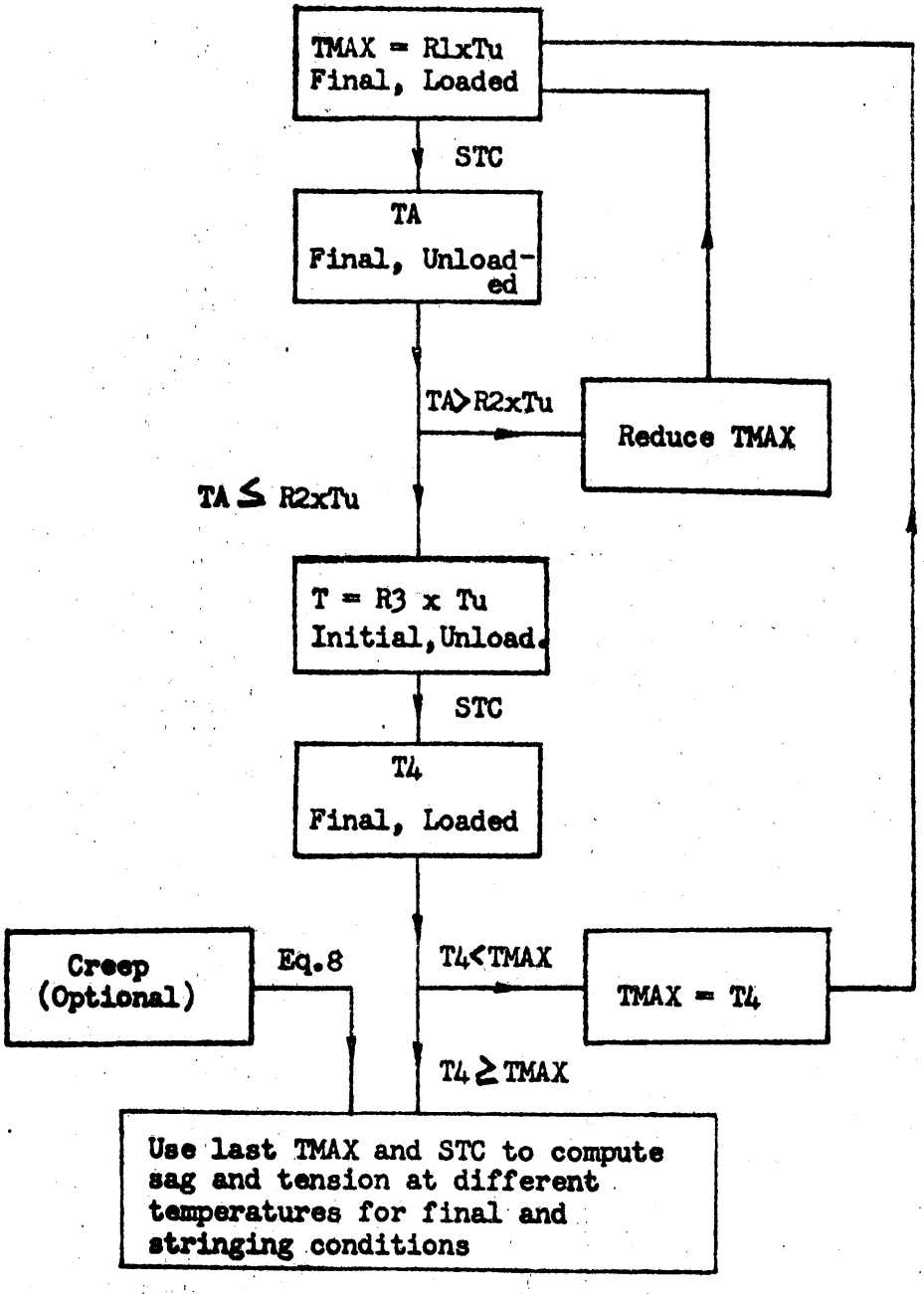


Fig. 3 Abbreviated Flow Diagram for Sag-Tension Computation

EXHIBIT 2

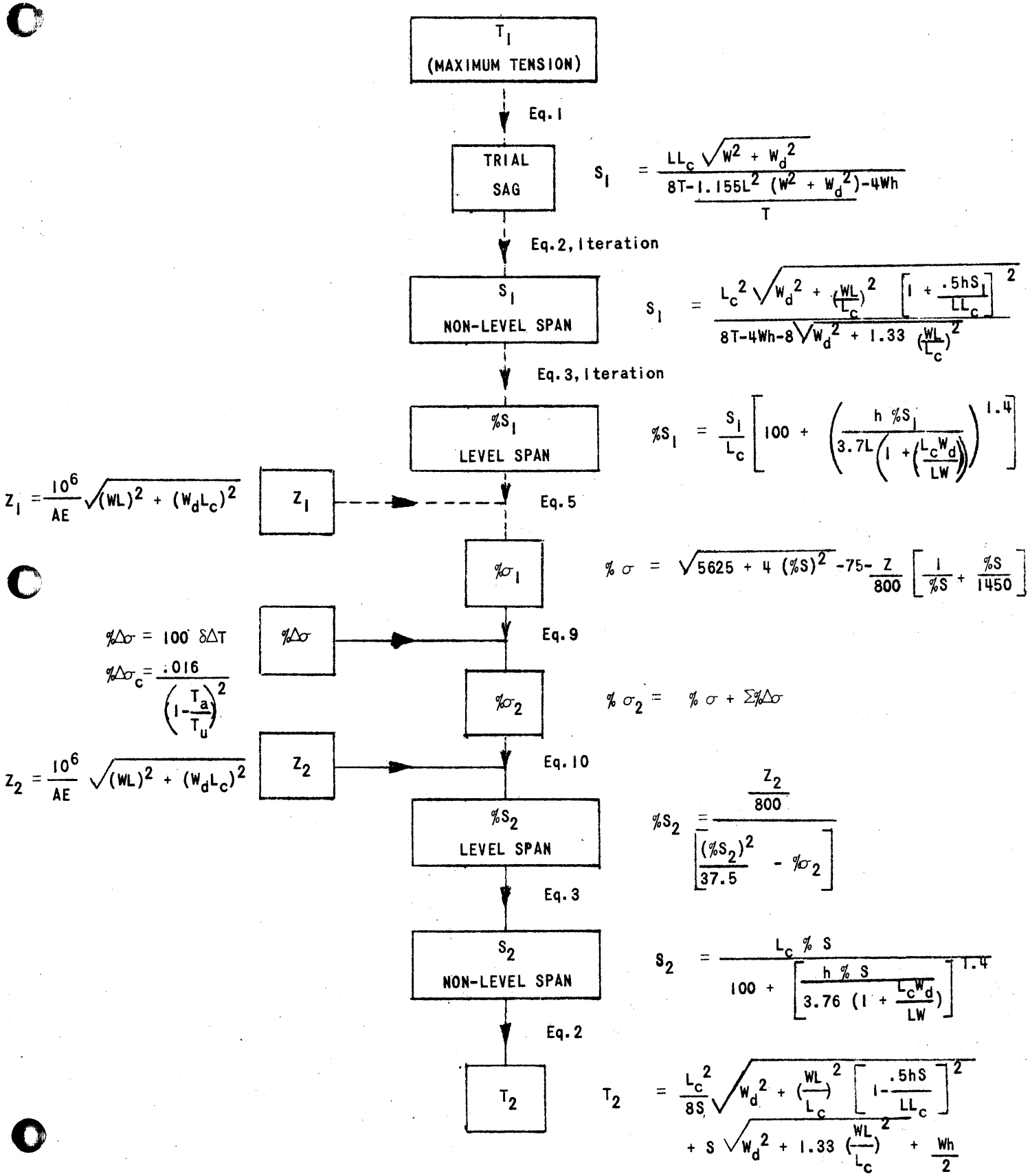


FIGURE 2 SIMPLIFIED FLOW DIAGRAM OF STC

EXHIBIT 3

IBM DATA FORMAT SHEET-SAG AND TENSION PROGRAM

SHEET 1 OF 1

	1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80
CARD 1	STEARNS-ROGER CORP. SAG AND TENSION PROGRAM E13
CARD 2	JOB NO. QL-1282 CUSTOMER COMMON MELTING PROJ. EXHIBIT 4
CARD 3	SUBJECT SAMPLE CALCULATION - MEDIUM SPAN BY J.R. SMITH DATE 7/7/66

30

CONDUCTOR DATA (1-CARD)

	X TYPE	X SIZE	X STRAND	DIAMETER	AREAC	▼ AREAS	AREAL	WEIGHT	ULT. TENSION	MAX. TENSION	+ CRPK	■ SPN
CARD 4	AW	248MCM	19/9	.572	.1954	.1525	.0429	.5658	34290.0	24000.0	.5	5

	■ NO.	LENGTH	EL-HI	EL-LO	W. PRESS.	ICE THK.	I. TEMP.	INC.	F. TEMP.	RULE1	RULE2	RULE3	X LOAD	■ *
CARD 5	1	1200.0	135.0	125.0	8.0	.50	0.0	1.0	120.0	60.0	25.0	35.0	HEAVY	1

CONDITIONS AND SPAN DATA (1-CARD)

PUNCH ONLY CARDS 1 THRU 5

* LOADING CODE HEAVY = 1, MEDIUM = 2, LIGHT = 3

\+ CREEP CONSTANT STEEL AND COPPER = .5, ACSR = 1.0, ALL ALUMINUM = 1.5

▼ AREAS CANNOT BE ZERO - IF NONE, ENTER .0001

■ FIXED POINT (WHOLE NUMBERS)

X ALPHABETIC CHARACTERS
ALL CHARACTERS TO BE RT. HAND JUSTIFIED

EXHIBIT 4

FORM E13.1

STEARNS-ROGER CORP.
JOB NO. DL-1282 CUSTOMER COMMON MEETING
SUBJECT SAMPLE CALCULATION - MEDIUM SPAN

SAG AND TENSION PROGRAM EI3
PROJ. EXHIBIT 5
BY J.K. SMITH DATE 7/7/66

SPAN	LENGTH	EL-HI	EL-LO	COND	SIZE	STRAND	WEIGHT	AREA	DIA.	LOAD	TENSION
1	1200.0	135.0	125.0	AW	248MCM	19/9	.5658	.1954	.572	HEAVY	13145.7

TEMPERATURE DEGREES	TENSION	FINAL SAG	LOW PT.	TENSION	STRINGING SAG	LOW PT.
0.0	13171.5	21.528	542.9			
0.0	8570.6	7.423	473.9			
10.0	8330.9	7.752	477.4			
20.0	8095.8	8.094	480.9			
30.0	7865.7	8.449	484.3			
40.0	7640.7	8.818	487.6			
50.0	7421.2	9.200	490.8			
60.0	7207.4	9.596	494.0			
70.0	6999.5	10.005	497.0			
80.0	6797.6	10.427	500.0			
90.0	6601.9	10.862	502.9			
100.0	6412.6	11.308	505.7			
110.0	6229.7	11.766	508.4			
120.0	6053.4	12.235	511.0			

EXHIBIT 5

STEARNS-ROGER CORP.
JOB NO. OL-1282 CUSTOMER COMMON MEETING
SUBJECT SAMPLE CALCULATION - MEDIUM SPAN

SAG AND TENSION PROGRAM E13
PROJ. EXHIBIT 6
BY J.R. SMITH DATE 7/7/66

SPAN	LENGTH	EL-HI	EL-LO	COND	SIZE	STRAND	WEIGHT	AREA	DIA.	LOAD	TENSION
1	1200.0	135.0	125.0	AW	248MCM	19/9	.5658	.1954	.572	HEAVY	13145.7

TEMPERATURE DEGREES	FINAL TENSION	FINAL SAG	FINAL LOW PT.	STRINGING TENSION	STRINGING SAG	STRINGING LOW PT.
0.0	13145.7	21.579	543.0			
0.0	8570.6	7.423	473.9	10118.5	5.695	451.1
10.0	8330.9	7.752	477.4	9855.6	5.948	454.9
20.0	8095.8	8.094	480.9	9595.9	6.213	458.8
30.0	7865.7	8.449	484.3	9339.6	6.489	462.5
40.0	7640.7	8.818	487.6	9087.0	6.778	466.3
50.0	7421.2	9.200	490.8	8838.3	7.079	469.9
60.0	7207.4	9.596	494.0	8593.7	7.393	473.5
70.0	6999.5	10.005	497.0	8353.5	7.720	477.1
80.0	6797.6	10.427	500.0	8118.0	8.061	480.6
90.0	6601.9	10.862	502.9	7887.3	8.415	484.0
100.0	6412.6	11.308	505.7	7661.9	8.782	487.3
110.0	6229.7	11.766	508.4	7441.9	9.163	490.5
120.0	6053.4	12.235	511.0	7227.5	9.558	493.7

UPLIFT AT LOWER SUPPORT OCCURS WHEN BOTH SAG AND LOW PT ARE ZERO

EXHIBIT 6

STEARNS-ROGER CORP.
JOB NO. OL-1282 CUSTOMER COMMON MEETING
SUBJECT SAMPLE CALCULATION - MEDIUM SPAN

SAG AND TENSION PROGRAM E13
PROJ. EXHIBIT 7
BY J.R. SMITH DATE 7/7/

SPAN	EL-HI	EL-LO	COND	SIZE	STRAND	WEIGHT	AREA	DIA.	LOAD	TENSION
1	1200.0	135.0	125.0	AW	248MCM	19/9	.5658	.1954	.572	HEAVY 131[illegible].1

TEMPERATURE DEGREES	FINAL TENSION	FINAL SAG	FINAL LOW PT.	STRINGING TENSION	STRINGING SAG	STRINGING LOW PT.
0.0	13184.9	21.501	542.8			
0.0	8570.6	7.423	473.9	10127.9	5.686	450.9
10.0	8319.6	7.768	477.6	9852.4	5.951	455.0
20.0	8073.7	8.127	481.2	9580.4	6.229	459.0
30.0	7833.4	8.501	484.8	9312.2	6.519	463.0
40.0	7598.8	8.889	488.2	9048.1	6.823	466.8
50.0	7370.2	9.293	491.6	8788.3	7.141	470.7
60.0	7147.9	9.711	494.9	8533.1	7.473	474.4
70.0	6932.1	10.143	498.0	8282.8	7.820	478.1
80.0	6723.1	10.590	501.1	8037.8	8.181	481.7
90.0	6521.0	11.049	504.1	7798.2	8.557	485.3
100.0	6325.9	11.522	507.0	7564.5	8.948	488.7
110.0	6137.9	12.006	509.8	7336.8	9.354	492.1
120.0	5957.1	12.502	512.4	7115.5	9.774	495.3

UPLIFT AT LOWER SUPPORT OCCURS WHEN BOTH SAG AND LOW PT ARE ZERO

FINAL VALUES INCLUDE 10 YEAR CREEP

EXHIBIT 7

STEARNS-ROGER CORP. SAG AND TENSION PROGRAM E13
JOB NO. OL-1282 CUSTOMER COMMON MEETING PROJ. EXHIBIT 8
SUBJECT SAMPLE CALCULATION - MEDIUM SPAN BY J.R. SMITH DATE 7/7/66

SPAN	LENGTH	EL-HI	EL-LO	COND	SIZE	STRAND	WEIGHT	AREA	DIA.	LOAD	TENSION
1	1200.0	135.0	125.0	AW	248MCM	19/9	.5658	.1954	.572	HEAVY	13145.7

TEMPERATURE DEGREES	FINAL TENSION	FINAL SAG	FINAL LOW PT.	STRINGING TENSION	STRINGING SAG	STRINGING LOW PT.
0.0	13171.5	21.528	542.9			
0.0	8570.6	7.423	473.9	10118.5	5.695	451.1
10.0	8330.9	7.752	477.4	9855.6	5.948	454.9
20.0	8095.8	8.094	480.9	9595.9	6.213	458.8
30.0	7865.7	8.449	484.3	9339.6	6.489	462.5
40.0	7640.7	8.818	487.6	9087.0	6.778	466.3
50.0	7421.2	9.200	490.8	8838.3	7.079	469.9
60.0	7207.4	9.596	494.0	8593.7	7.393	473.5
70.0	6999.5	10.005	497.0	8353.5	7.720	477.1
80.0	6797.6	10.427	500.0	8118.0	8.061	480.6
90.0	6601.9	10.862	502.9	7887.3	8.415	484.0
100.0	6412.6	11.308	505.7	7661.9	8.782	487.3
110.0	6229.7	11.766	508.4	7441.9	9.163	490.5
120.0	6053.4	12.235	511.0	7227.5	9.558	493.7

UPLIFT AT LOWER SUPPORT OCCURS WHEN BOTH SAG AND LOW PT ARE ZERO

FINAL VALUES INCLUDE 10 YEAR CREEP

EXHIBIT 8

STEARNS-ROGER CORP. SAG AND TENSION PROGRAM C13
JOB NO. OL-1282 CUSTOMER COMMON MEETING PROJ. EXHIBIT 9
SUBJECT SAMPLE CALCULATION - LONG SPAN BY J.R. SMITH DATE 7/7/66

SPAN	LENGTH	EL-HI	EL-LO	COND	SIZE	STRAND	WEIGHT	AREA	DIA.	LOAD	TENSION
5	9999.9	1000.0	1000.0	ALWD	610MCM	37/8	1.3980	.4798	.899	HEAVY	37454.7

TEMPERATURE DEGREES	FINAL TENSION	FINAL SAG	FINAL LOW PT.	STRINGING TENSION	STRINGING SAG	STRINGING LOW PT.
0.0	37767.9	919.896	4999.9			
0.0	21049.7	890.961	4999.9	21290.7	879.312	4999.9
30.0	20961.3	895.314	4999.9	21198.8	883.715	4999.9
60.0	20874.3	899.648	4999.9	21108.2	888.099	4999.9
90.0	20788.5	903.964	4999.9	21019.0	892.465	4999.9
120.0	20703.9	908.262	4999.9	20931.1	896.811	4999.9

UPLIFT AT LOWER SUPPORT OCCURS WHEN BOTH SAG AND LOW PT ARE ZERO

FINAL VALUES INCLUDE 10 YEAR CREEP

EXHIBIT 9

ACF INDUSTRIES
INCORPORATED
ALBUQUERQUE DIVISION
MANAGEMENT SYSTEMS

AUTOSPOT III POSTPROCESSOR ORGANIZATION OUTLINE

COMMON MEETING
July 7, 1966
Denver Hilton Hotel Denver, Colorado

Prepared by: J. J. Hayden
J. J. Hayden
Engineer

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Numerical Control (N/C) is the operation of a device under control of numerical data (usually recorded in punched tape). It is the process of controlling machinery by automatic means rather than by manual means. An example would be the controlling of a drill press with the data recorded in a punched tape.

1.2 AUTOSPOT (AUTOmatic System for POsitioning Tools) III is a general purpose computer program, developed by IBM for use with the 1620 Monitor I System, designed to aid in preparing data primarily for point-to-point machine tools. The program also provides limited contouring capabilities.

1.3 This report presents a general approach to organized postprocessor development for the AUTOSPOT III program. The approach is centered around a structural outline with an emphasis on modularization and standardization. This outline is demonstrated in the form of a postprocessor input module (AUTOIT) that allows for the expansion of the AUTOSPOT III input language and a postprocessor program for the 3 BHT-B Burgmaster Turret Drilling Machine.

2. AUTOSPOT III POSTPROCESSOR OUTLINE

2.1 Postprocessor Modularization

The functions of a postprocessor are divided into five major modules or sections. Similarly, each major section can be divided into various sub-sections. Modularization can best be attained using the FORTRAN II-D language; therefore, whenever possible, this language will be used in coding the various sections of the postprocessor.

2.1.1 Control Section. The Control Section initiates execution of all five major sections and maintains control of the flow of data in the postprocessor. This section should be very near the same for all postprocessors.

2.1.2 Input Section. The Input Section reads input for the postprocessor and transfers it into a buffer area in a prescribed format. This section can be used to expand the AUTOSPOT language through the use of the POSTPR/ statement. (See Section 3.)

2.1.3 Geometry Section. The Geometry Section computes all of the axes movements and converts this data to the machine tool coordinate system for the Output Section. Any adjustments to the coordinate data will also be provided in this section. An example would be the adjustments required for coordinates that are outside the machine tool limits.

2.1.4 Auxiliary Section. The Auxiliary Section executes action specified by auxiliary instructions and develops auxiliary command blocks for the Output Section. An example would be a value specified for scaling a plot or an auxiliary command generated to set a particular machining mode. The tables used by this section should be standardized as much as possible.

2.1.5 Output Section. The Output Section provides for editing of all output information. The output data will be printed and plotted as it is received from the various other sections. The information to be punched in the control tape will be stored on disk until processing of all of the AUTOSPOT III output is completed. If no unrecoverable errors have been detected, the information will then be punched in the control tape.

2.2 Standard Common

In order to provide interchange of data between the sections, certain internal data (variables, parameters, switches, etc.) will be part of common storage. Common will be divided into areas for each major section of the postprocessor. The organization of the common area along with the standard sections will simplify communications between AUTOSPOT III postprocessor programmers.

2.3 System Description

2.3.1 The Control Section initalizes the postprocessor then calls the Input Section to provide information in the input buffer area. Control will identify the record type and determine the section required to initiate processing of the information.

2.3.2 The Geometry Section is called to process coordinate information. A discrete block of associated data (machine tool coordinate, spindle specification, auxiliary function, etc.) is generated for each coordinate record input. Once a block of data has been generated, its relationship to neighboring blocks of data is determined. (An example would be if there is a minimum or maximum departure required from the previous set of coordinates.) If the relationship is acceptable, the data is printed, plotted, and stored on disk to be punched in tape following processing of the entire AUTOSPOT III output. If the relationship is unacceptable, suitable correction is obtained if possible, and the proper diagnostics printed. If suitable correction is impossible processing will continue but the control tape will be eliminated.

2.3.3 The Auxiliary Section will process the non-motion instructions in the input buffer. Certain auxiliary instructions will simply set program mode and parameters, others will execute special logic, and still others will develop auxiliary command blocks. Thus the Auxiliary Section will be a set of calls to sub-sections designed to react for a specific instruction.

2.3.4 The Output Section will be called by the Geometry and Auxiliary Sections to plot and print the output information and to store on disk the information that is to be punched in the control tape. When processing of the AUTOSPOT III output is complete, Control will call Output to punch the information stored on disk in a control tape, if no unrecoverable errors have occurred.

3. AUTOIT (AUTOspot Input Translator)

3.1 General

AUTOIT is an input subroutine for AUTOSPOT III Postprocessors. The program reads AUTOSPOT III output and translates it into a prescribed form. Also, the subroutine provides a method for increasing the AUTOSPOT III language through the use of the POSTPR/ statement. The program is written in SPS II-D and can be used by both SPS II-D and FORTRAN II-D programs.

3.2 POSTPR/ Statement

AUTOIT will accept a maximum of 79 characters following the slash in the POSTPR/ statement. A combination of postprocessor words, fixed point numbers, floating point numbers, and special characters can follow the slash. The postprocessor words must be defined in the table provided in AUTOIT. The fixed point numbers can be a maximum of 4 digits and the floating point numbers can be a maximum of 8 digits. The special characters =(,*$) can be used to separate the above three type of fields.

3.3 Subroutine Description

The output from the AUTOSPOT III program consists of variable length records written on disk. The contents of these records are processed as follows:

(a) Fixed Point Numbers. The fixed point numbers are from two digits to six digits in length. The fixed point numbers in each record are converted to four digit fixed point numbers and stored sequentially in an array provided by the users. If the number is larger than four digits, it is separated into two fixed point numbers, the four lower order digits stored as one number and the remaining digits as a separate number.

(b) Floating Point Numbers. The floating point numbers have either eight digit or thirteen digit mantissas. All of the floating point numbers are converted to eight digit mantissa floating point numbers and stored sequentially in an array provided by the user.

(c) Alphameric Data. Alphameric data is separated into four digit fixed point fields and stored sequentially in the same array used by the fixed point numbers.

(d) Postprocessor Records. The floating point and fixed point numbers are stored as described above. Each postprocessor word is decoded according to the table provided in the subroutine. The number code corresponding to each postprocessor word is stored sequentially in an array provided by the user. If the postprocessor word is PARTNO, the alphameric information following the slash in the POSTPR/ statement is stored in an array provided by the user.

3.4 Linkage

3.4.1 FORTRAN II-D

CALL AUTOIT (IDSKRD, IFIX(1), FLOAT(1), IPP(1), IPTNO(1))

(a) IDSKRD = 0 Initialize AUTOIT to read first AUTOSPOT record.
= 1 Continue processing the next AUTOSPOT record.

(b) IFIX (1) The first field of the fixed point array. Minimum array size is 10.

(c) FLOAT (1) The first field of the floating point array. Minimum array size is 10.

(d) IPP (1) The first field of the postprocessor codes array. Minimum array size is 10.

(e) IPTNO (1) The first field of the array provided for the PARTNO statement. The minimum array size is 38.

3.4.2 SPS II-D

(a) The following statement must be entered in the user's program:

(1) CALL LOAD, AUTOIT, @@SUB

This statement loads AUTOIT into memory from disk and must precede the calling sequence.

(2) @@SUB DSS SIZE

@@SUB is the location where AUTOIT is to be loaded and must be defined in an even numbered location. SIZE is the length of the subroutine. It is variable because of the additions that may be made to the postprocessor word table.

(b) The following sequence of instructions are used to "call" AUTOIT:

```
BTM @@SUB, *+ 11
DSA IDSKRD, IFIX1, FLOAT1, IPP1, IPTNO1
```

where the above fields are defined as follows:

```
IDSKRD      DC      4, 0
IFIX1       DSB     10, 4
IFLOAT1     DSB     10, 6
IPP1        DSB     4,10
IPTNO1      DSB     4,38
```

3.5 Postprocessor Words

Each entry to the postprocessor word table requires six alphabetic and four numeric characters. Additional postprocessor words can be added to AUTOIT. These additions must be entered in the postprocessor word table following the remark:

```
*     ENTRIES TO THE POSTPROCESSOR WORD TABLE SHOULD BEGINE HERE
```

A postprocessor word of six characters such as MACHIN with a number code such as 0002 would be entered as follows:

```
DAC    6, MACHIN
DC     4, 0002
```

A postprocessor word of less than six characters such as RAIL with a number code such as 0010 would be entered as follows:

```
DAC    2, XX
DAC    4, RAIL
DC     4, 0010
```

4. AUTOSPOT III POSTPROCESSOR EXAMPLE

The following is an example of an AUTOSPOT III Postprocessor that was written using the AUTOSPOT Postprocessor Organization Outline described previously. The postprocessor is written for the 3 BHT-B Burgmaster Turret Drill.

4.1 Postprocessor Description

4.1.1 AUTOSPOT Part Program Requirements

(a) Specification Statements

All of the specification statements can be used except the following:

(1) BKC

(2) SAFE

(b) Major, Minor, and Geometry Statements

All of the point-to-point major, minor, and geometry statements can be used. The following POSTPR/ statements can be used:

(1) POSTPR/PARTNO, NNNNNN
This statement specifies the part number

(2) POSTPR/MACHIN, BURG
This statement specifies the Burgmaster Postprocessor to be used.

(3) POSTPR/PPLOT, START, XX.X, YY.Y, SCALE, SS.S
This statement initiates plotting at the point (XX.X, YY.Y) at a scale of SS.S. If this statement is omitted from the part program, no plot will be drawn.

(4) POSTPR/PRESET
This statement must be entered in the part program if the computed Z and W values are to be output. If this statement is omitted from the part program, a Z of 2 and W of 1 will be output for each individual pattern generated.

(5) POSTPR/RAIL, XX.X
This statement specifies the rail height XX.X. If this statement is omitted from the part program, a rail height of 39 inches is assumed.

4.1.2 Spindle Selection. The spindle (1-8) will be selected by the SP(N) statement. The following statement would cause spindle 6 to be selected.

TAP,20,SP(6).....

4.1.3 Mode Selection. The machining operations tap and bore generate the mode E4. All other machining operations generate the mode E3.

4.1.4 Diagnostics

Error Number	Cause
1	This error occurs when the machine tool axis limits (X00000-600000, Y-00000-480000) are exceeded. The point in question is plotted and listed but is eliminated from the control tape.
2	This error occurs when the minimum movement along the X or Y axis is not satisfied. That is, movement along both the X and Y axes must be zero or greater than or equal to .100. If this error occurs, the point in question will be stored and then processed when it is possible to do so without error. If it is not processed before the next spindle change, a "dummy" (X, Y) movement of .200 will be inserted followed by the point in question.
3	Illegal input code.
4	Illegal postprocessor word.

5	Illegal spindle number.
6	Storage table for minimum movement points is exceeded. Maximum is 10.
7	Illegal machining operation.

The error messages are output on the printer and processing continues.

4.2 Input Requirements. The input to the program is the output from the AUTOSPOT III program. The input is read from disk using the AUTOIT subroutine.

4.3 Output Requirements

4.3.1 Listing. See Figure 1.

4.3.2 PLOT. See Figure 2.

4.4 Control Tape

	Output Format	Remarks
1.	XnnnnnYnnnnnDn ≠	This format is used when an X, Y coordinate and a new turret number are to be output.
2.	XnnnnnYnnnnn ≠	This format is used to output X, Y coordinates without a turret number.
3.	ZnnnnnWnnnnnEn ≠	This format is used when a Z and W value is to be output and a change in the mode of operation has occurred.
4.	ZnnnnnWnnnnn ≠	This format is to be used when a Z and W value is to be output without a change in the mode of operation.
5.	ZW ≠	This format is used to output a Z and W value that is identical to the previous Z and W values output and there is no change in the mode of operation.
6.	M ≠	Stop command.

Ten channel five punches will precede each output block containing a turret change (format 1). Format 3 will be used any time a mode command is output. If a part number is input, it is imaged in the control tape preceding and following output data.

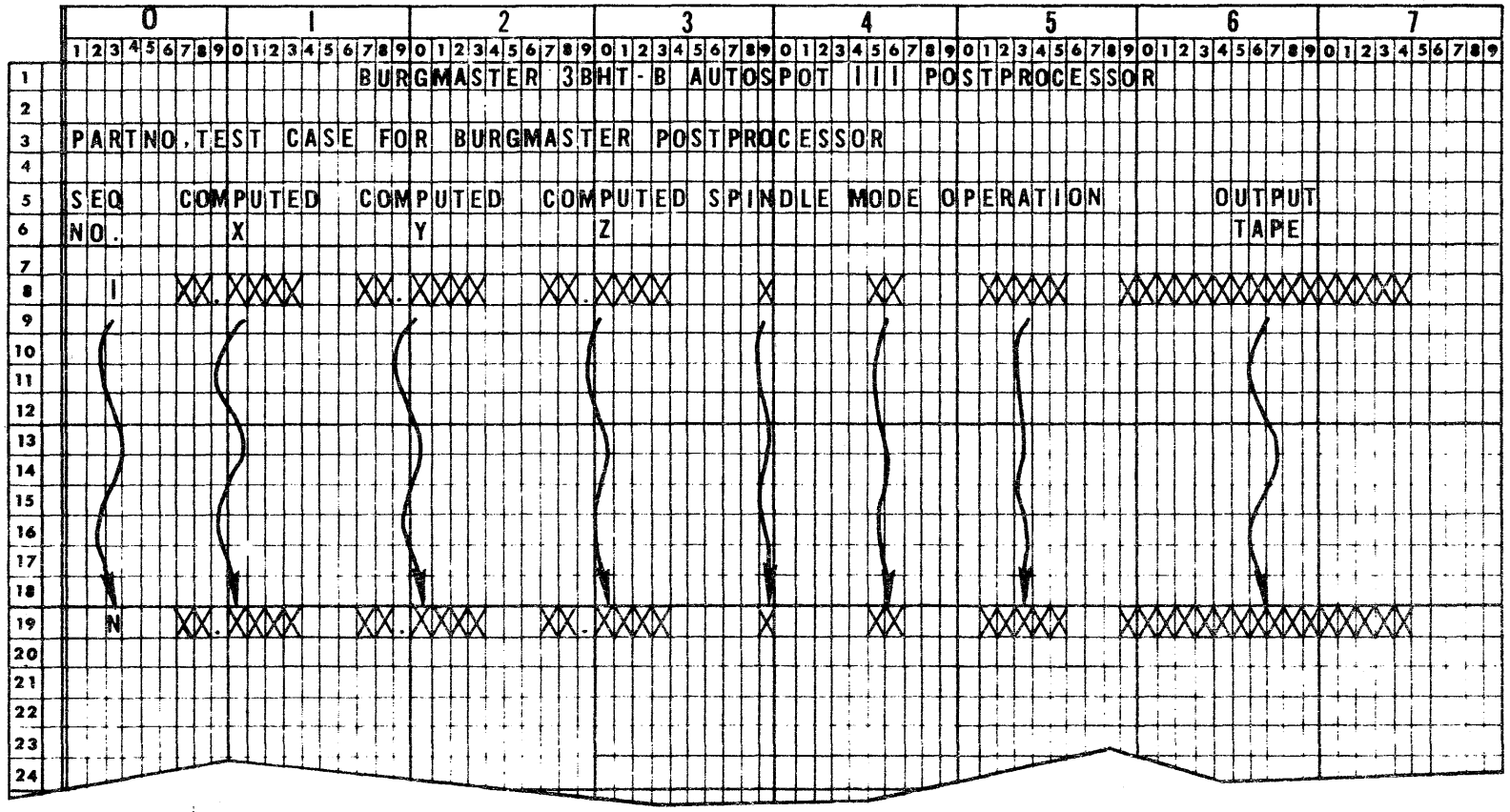


Figure 1. SAMPLE LISTING

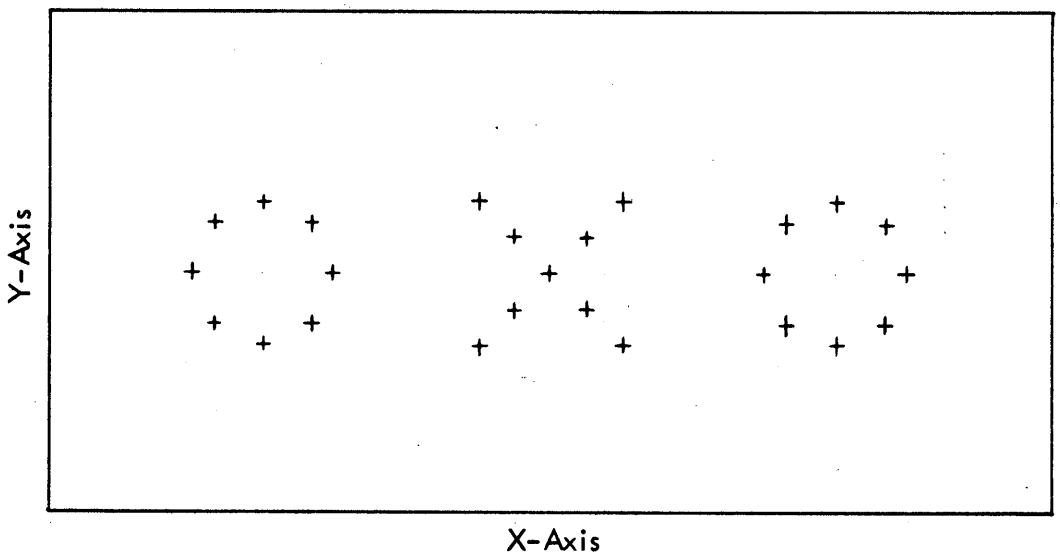


Figure 2. SAMPLE PLOT

DEVELOPMENT OF A PROGRAM TO DESIGN ME-LTV EVAPORATORS FOR SEA WATER CONVERSION

by

David D. Kays
Stearns-Roger Corporation
Denver, Colorado

presented at the

Western Region Summer Meeting
of
COMMON

July 6, 7, and 8, 1966

Denver, Colorado

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

The increasing need for potable and ultrapure water, created by urbanization, industrialization and population growth has caused local and regional shortages to arise. The extensive planning and forecasting carried on by several national governmental agencies and international organizations has indicated that shortages would become limiting to at least economic growth of extensive geographic areas in the period beginning not later than 1980. One source of water, which matches the population concentrations in the coastal areas of many nations, is the sea. Desalting of this water for human and industrial purposes is not entirely new, but what is new is the attempt to do this on a large and relatively economical scale.

The development of storage for and transportation of large quantities of imported water requires decades of planning and construction, as well as large capital investments. Therefore the U. S. Government has undertaken a program of desalting technology development and demonstration to circumvent some of the costs and the time lag. This program aims to (1) decrease the cost of desalting water; (2) develop the experience and technology for many sizes of plants and processes; and (3) provide new water sources close to the points of need in a relatively short period of time. The program is under the direction of the Office of Saline Water, U. S. Department of the Interior.

Stearns-Roger Corporation was named Management and Operations Contractor in April 1961, for the first production size Sea Water Desalting Demonstration Plant built by the Office of Saline Water, at Freeport, Texas. For the past five years, the purposes of plant operations have been, the continuous improvements in performance, and the development of reliable design data and hardware. About two and one half years ago, the operations reached the stage where optimization became important, and enough data was available to consider computer design programs. Stearns-Roger initiated advanced hardware design efforts which have resulted in a presently active, expansion program at the Freeport Plant. Several important process problems were solved which resulted in superior performance of the ME-LTV process, creating a high demand for proposals of completely designed plants to compete with the previously popular Multi-Stage Flash process. This situation has resulted in the Computer Programs presented in this paper.

II. THEORY AND PROCESS

The Freeport Plant is an initial design of a Multiple Effect-Falling Film Sea Water Evaporator. It is a several-fold expansion of Multiple Effect

chemical solution evaporators. The process has a successful industrial history extending over more than fifty years. The plant represents an extension of twelve effects and has achieved a major improvement in thermal economy. It also, has successfully demonstrated scale free operation using the highly effective falling film evaporation in long vertical tubes. The process name is derived from these characteristics, since ME-LTV is the representation commonly used for Multiple Effect-Long Tube Vertical sea water evaporators.

To fully identify the most useful sea water distillation form of the process, it is necessary to specify forward series feed, with regenerative feed heating. Many modes of operation have been investigated at the Freeport Plant, resulting in much simplification of the process and equipment. The advanced design equipment incorporating the simplifications, is now being installed at the Freeport Plant and is represented by Figure 1, Process Flow Diagram -- ME Falling Film Plant. This diagram is also self-explanatory of the forward, series fed, regeneratively heated feed, multiple effect evaporator. The diagram is also the process basis for the computer programs which follow. Symbols appearing on the diagram are listed in the appendix.

Prior to the establishment of the preliminary computer program, it was necessary to make detailed process calculations on an effect by effect basis, starting with Effect No. 1. These calculations were necessarily based on a number of assumed flows and conditions. It required approximately 24 man hours to complete a set using a desk calculator. If the results of the assumptions were not acceptable, it was necessary to adjust a flow rate, temperature, etc., and recalculate the entire set, just to bring the extraction ratio, duty, or some other result into an acceptable range. Arithmetic transposition errors so frequent in repetitive calculations have the effect of invalidating all successive calculations in a set, since the performance of Effect n + 1 is totally dependent on the performance calculated for Effect n. This method of design calculations was a severe limitation on the consideration of alternates for any set of proposal conditions, and thus precluded most optimization other than the judgment of an experienced engineer. Geometry calculations had to follow a final set of acceptable process calculations.

In the ME-LTV system, the most important considerations which are under the designer's control are, to divide the total temperature spread between heating steam and heat sink in some manner, and to fix the extraction ratio. These two basic considerations must result in a plant which operates within boundary values fixed by the site, such as the production rate, sea or brackish water supply, available land area, and available energy sources. A more concise statement of the independent design variables is given in Table 1.

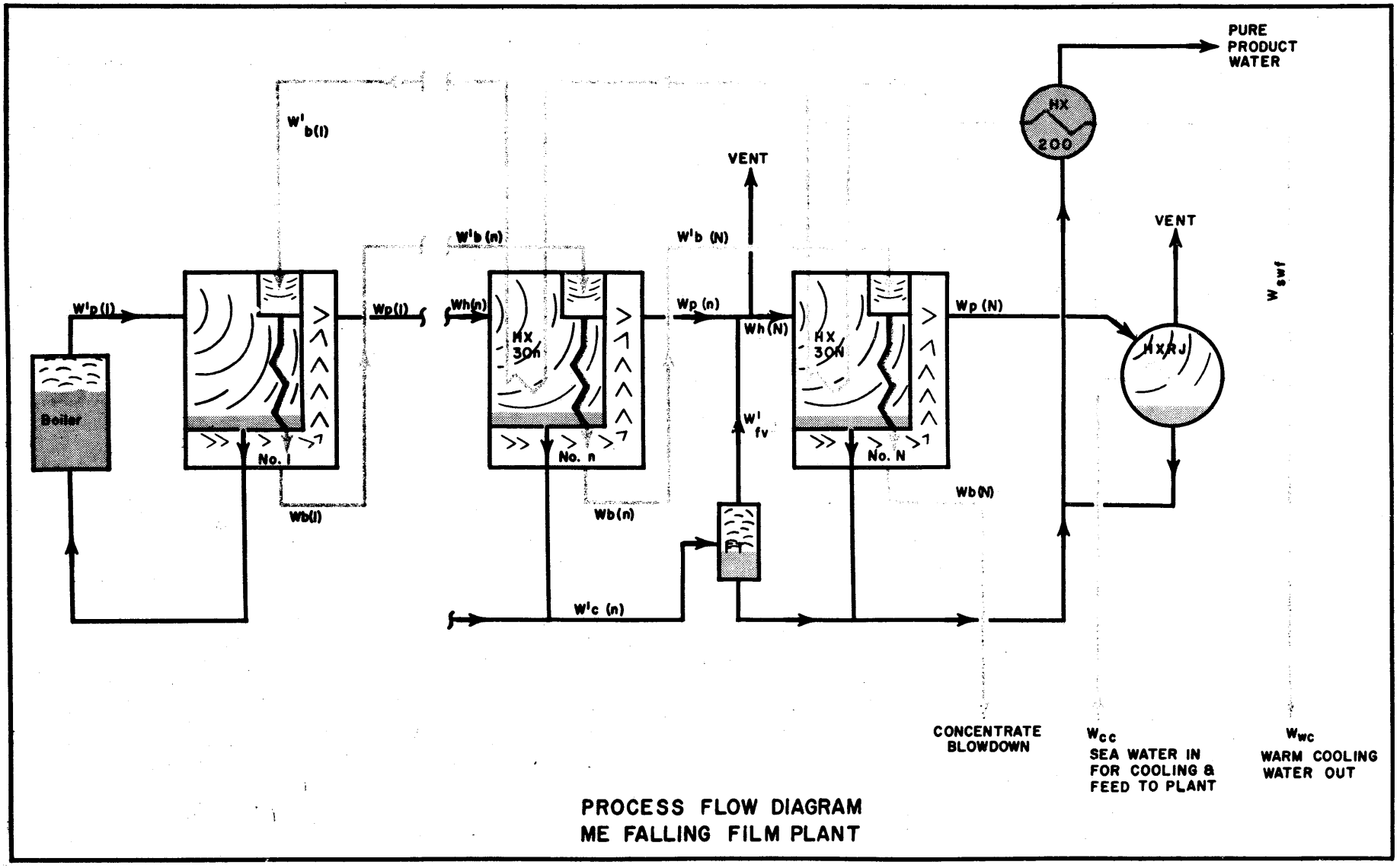


FIGURE 1. PROCESS FLOW DIAGRAM ME FALLING FILM PLANT

VARIABLES UNDER THE DESIGNERS CONTROL	DESIRED RESULTS
1. Vapor Temperature Profile	Principal Result
2. Sea Water Preheating Profile	Minimum Cost of Water
3. Extraction Ratio	Subordinate Results
4. Heating Steam Temperature	1. Minimum Heat Transfer Surface
5. Heat Rejection Temperature	2. Minimum Pumping Energy Requirement
6. Equipment Geometry	3. Optimum Thermal Economy
	4. Optimum Geometry
	5. Minimum Energy Input
	6. Minimum Capital Cost
	7. Optimum Plant Life

TABLE 1. IMPORTANT VARIABLES AND DESIRED RESULTS

Most of these process variables can be changed arbitrarily by the designer by varying such things as the temperature profiles, the number of effects, the approach of stream temperatures in the heat exchangers, etc., within the boundary limits previously mentioned. The costs of fuel and money are generally fixed by the user and are related to the site. There are a greater number of dependent variables which change each time one of the independent variables is changed. This illustrates the complexity of the design problem whose ultimate goal is the minimum cost of water, for most situations. Other desired end results are possible for less common situations, and are listed in Table 1.

Specifically, it was desired that a computer program be developed which would take a given set of terminal conditions, capital and fuel costs and determine the optimum plant characteristics in accordance with Table 2.

The lack of adequate design correlations and reliable cost data still hamper all phases of the effort to develop a complete program. These two problems are receiving much attention at present, with particular emphasis on cost records for the module now being installed at Freeport.

III. THE PRELIMINARY PROGRAM

The development of simplified equations, from thermodynamic considerations, was followed by a check of their accuracy. This check was made by comparing the predicted performance to actual performance of the Freeport Plant. These equations and assumptions are contained in a previous paper by the author (Reference 1). They form the basis of the preliminary program depicted by Figure 2. This program requires considerable input by the designer, but does eliminate the problem of arithmetic and transposition errors. It runs on the 1620 IBM computer in slightly less than two minutes.

To prepare the initial set of input data requires approximately two hours of an experienced engineer's time. The required input information is listed in Table 3. Additional input, where variations to be investigated are set forth, can be established in less than a half hour. Manipulations, based on results of the initial sets of input data, are made rapidly by repunching only a few cards in a deck. Thus, it is possible consider eight to twelve alternatives in a period of one day; if the computer is available.

The output of this program is limited to the most critical portions of the process and geometry portions of the desired end results. It has only one process loop, which adjusts the preheating temperature profile to increase or decrease the thermal performance around a target value, using the input number of effects. Even so, it provides the engineer with the necessary end results to make a logical judgment optimization

1. The Optimum Process Parameters

   a. N, the number of effects and operating temperature profiles.

   b. The duties, temperature differences and heat transfer coefficients of all heat exchangers.

   c. The velocity, quantity, quality and location of all process stream flows, for the required production rate.

   d. The fuel, electricity and chemical requirements for the required production rate.

   e. The Net thermal efficiency; extraction ratio, feed water, heating steam, product, and waste brine quantities for the optimum plant process.

2. The Geometry of the Optimum Plant

   a. Heat exchanger dimensions.

   b. Vessel dimensions.

   c. Pump requirements; flow, head, driver horsepower.

   d. Vacuum equipment requirements; volume, pressures, cooling water and energy requirements.

3. Cost Data for the Optimum Plant

   a. Capital cost of the complete plant to battery limits.

   b. Actual product water cost, with the total broken down into operating, maintenance and fixed costs.

4. Sensitivity of the Selected Optimum Design

   a. To variations in fixed charge rates.

   b. To variations in energy costs.

TABLE 2. DESIRED OUTPUT FROM A COMPUTER PROGRAM

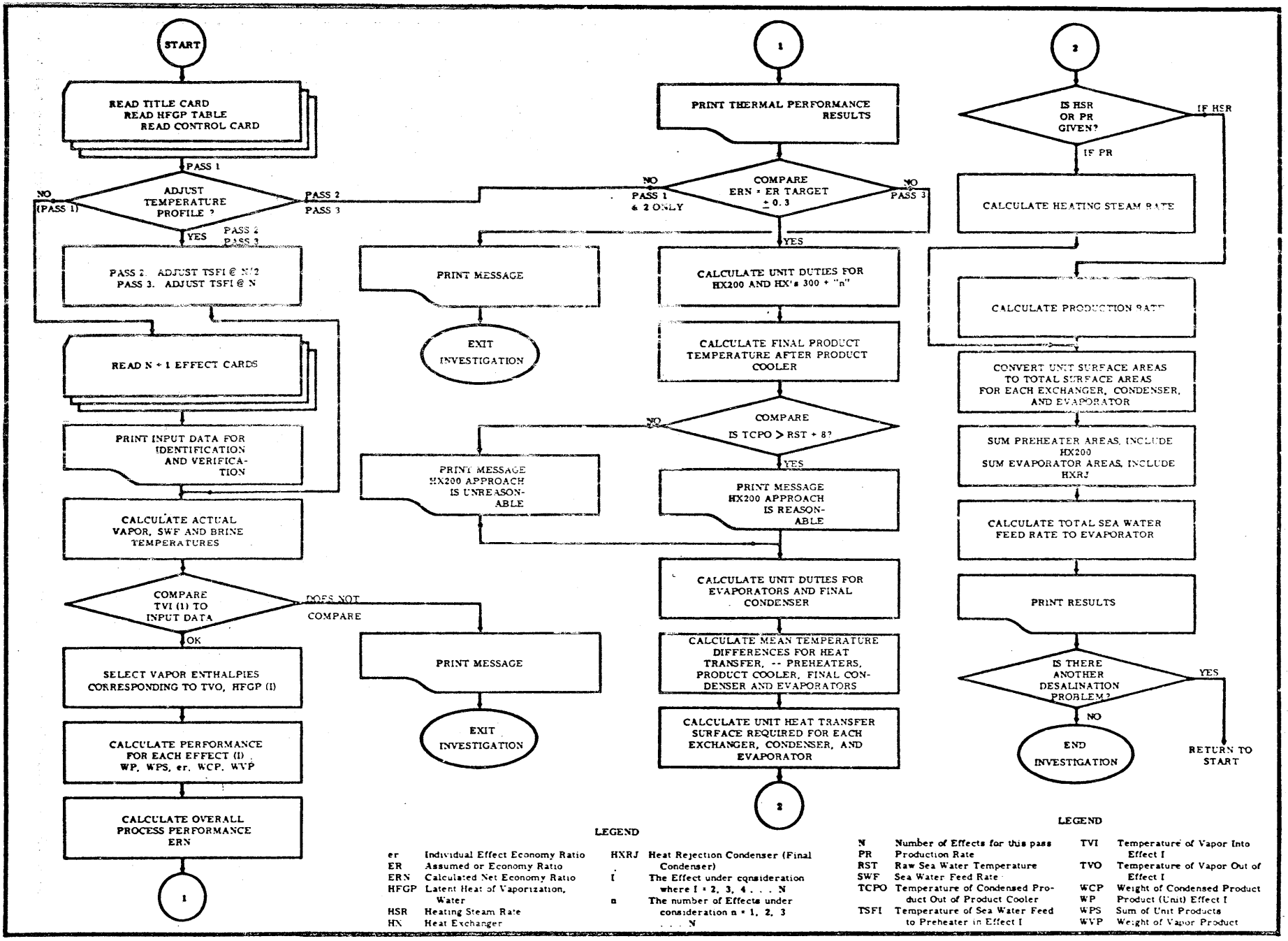


FIGURE 2. BLOCK DIAGRAM, PRELIMINARY PROGRAM

OVER-ALL CONTROL DATA	DATA FOR EACH INDIVIDUAL EFFECT (N + 1)
Number of Effects	Boiling Point Rise
Feed to Steam Ratio	Vapor to Vapor Temperature Difference
Target Economy Ratio	Sea Water Preheating Temperature Rise
Steam Inlet Temperature	Vent Loss
Heat Rejection Condensing Temperature	Heat Loss to Environment
Specific Heat of Brine	Boiler Feed Water Return Quantity
Sea Water Feed Inlet Temperature	Over-all HT Coefficient for Preheater
Vent Loss From Final Condenser	Over-all HT Coefficient for Evaporator
Raw Sea Water Temperature	Temperature Rise up to the tube due to pressure change
Temperature Rise of Sea Water in Final Condenser	
Heating Steam Rate or Production Rate	

TABLE 3. PRELIMINARY PROGRAM INPUT DATA REQUIREMENTS

as far as total required heat exchanger surface required for given production rate and thermal economy is concerned. Since this surface area accounts for approximately 50 percent of the total cost of a plant, very rough cost optimization can be manually accomplished. The cost of fuel and the applicable fixed charge rate broadly categorize the usable thermal economy ratios, as a matter of experience. For the reasons stated above, this preliminary program has been very useful and serves quite well as the starting point for heat exchanger and vessel design.

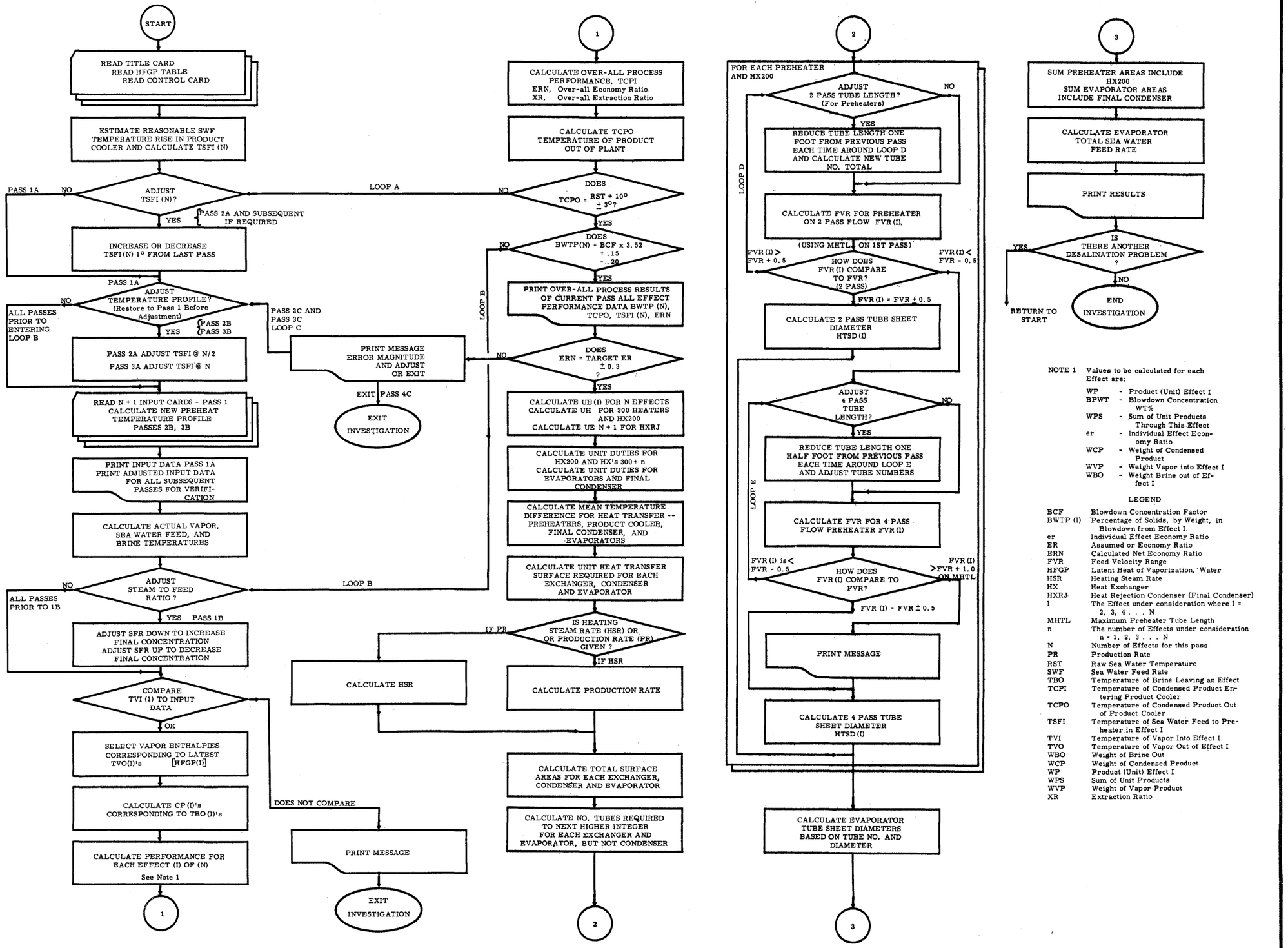
IV. THE INTERMEDIATE PROGRAM

This program is in the development stage. Additional equations have been written, and modifications to the existing equations have been developed. This program becomes possible as a result of the acquisition of sufficient operating data to develop temperature related curves for heat transfer coefficients and specific heats. These are empirical relationships limited to normal sea water feed and an extraction ratio of approximately 0.67 (2/3 pound of water extracted from each pound of feed). Minor variations of these two conditions can be tolerated without much error. The Block Diagram of Figure 3 illustrates the flow pattern for this program.

The program has five built in optimization loops. Three of these operate on process conditions to achieve (1) the optimum preheating profile, (2) the correct extraction ratio, and (3) the desired thermal economy. Two loops operate on geometry factors to provide (1) the correct number of passes and (2) tube length. The requirements for input data, given in Table 4 are considerably changed. The time consuming process variable inputs have been significantly reduced. The specification of broadly applicable geometry considerations as input have been increased.

The output of this program will largely fulfill the requirements listed in Table 2, for items 1, process and 2, geometry, with some exceptions. The principal process outputs still missing are (1) an optimized number of effects, and (2) electrical and chemical requirements. The geometry section will include all the desired information for the input design except, the vessel dimensions, pump heads and horsepowers, and vacuum equipment requirements. The items provided will be a large percentage of those required for final design and costing of a proposed plant. The entire program, with its reduced input data, will provide a much greater insight to the engineer for manual optimization, with little, if any, increase in the time required. This program is expected to be operational by September, 1966.

Computations with the intermediate program will provide a great deal of insight into the design of the costing step, and eventually the programmed



NOTE 1 Values to be calculated for each Effect are:

- WP - Product (Unit) Effect I
- BPWT - Blowdown Concentration WT%
- WPS - Sum of Unit Products Through This Effect
- er - Individual Effect Economy Ratio
- WCP - Weight of Condensed Product
- WVP - Weight Vapor into Effect I
- WBO - Weight Brine out of Effect I

LEGEND

BCF	Blowdown Concentration Factor
BWTP (I)	Percentage of Solids, by Weight, in Blowdown from Effect I
er	Individual Effect Economy Ratio
ER	Assumed or Economy Ratio
ERN	Calculated Net Economy Ratio
FVR	Feed Velocity Range
HFGP	Latent Heat of Vaporization, Water
HSR	Heating Steam Rate
HX	Heat Exchanger
HXRJ	Heat Rejection Condenser (Final Condenser)
I	The Effect under consideration where I = 2, 3, 4 . . . N
MHTL	Maximum Preheater Tube Length
n	The number of Effects under consideration n = 1, 2, 3 . . . N
N	Number of Effects for this pass.
PR	Production Rate
RST	Raw Sea Water Temperature
SWF	Sea Water Feed Rate
TBO	Temperature of Brine Leaving an Effect
TCPI	Temperature of Condensed Product Entering Product Cooler
TCPO	Temperature of Condensed Product Out of Product Cooler
TSFI	Temperature of Sea Water Feed to Preheater in Effect I
TVI	Temperature of Vapor Into Effect I
TVO	Temperature of Vapor Out of Effect I
WBO	Weight of Brine Out
WCP	Weight of Condensed Product
WP	Product (Unit) Effect I
WPS	Sum of Unit Products
WVP	Weight of Vapor Product
XR	Extraction Ratio

58

FIGURE 3. BLOCK DIAGRAM, INTERMEDIATE PROGRAM

OVER-ALL CONTROL DATA	DATA FOR INDIVIDUAL EFFECTS
Number of Effects	Boiling Point Rise
Initial Steam to Feed Ratio	Vapor to Vapor Temperature Difference
Target Economy Ratio	Sea Water Preheating Temperature Rise
Steam Inlet Temperature	Vent Loss
Heat Rejection Condensing Temperature	Heat Loss to Environment
Temperature Rise of Sea Water in Final Condenser	Boiler Feed Water Return Quantity
Target Preheater Brine Velocity Range	Temperature Rise up the tube due to pressure change
Inlet Sea Water Concentration	Evaporator tube diameter and gauge
Vent Loss From Final Condenser	
Raw Sea Water Temperature	
Heating Steam Rate	
or	
Production Rate	
Target Extraction Ratio	
Maximum Preheater Tube Length	
Evaporator Tube Length	

TABLE 4. INTERMEDIATE PROGRAM INPUT DATA REQUIREMENTS

optimization of the over-all plant, which is presented in the next section.

V. AN ULTIMATE PROGRAM

The ultimate program required for design of the optimum plant may be represented in a generalized form by Figure 4. The subordinate desired end results listed previously in Table 1, are not all provided, so some manipulation of the input, based on preliminary output will still be required for unusual situations. However, the principal desired result of minimum water cost will certainly be provided within the accuracy of the input data. The subordinate desired results of minimum surface area, optimum thermal economy, minimum capital cost, and optimum plant life, will be provided by the same design as minimum water cost in a very high percent of situations encountered. Sensitiveness of the optimum design to fuel cost changes and interest or fixed charge rates can be investigated very easily by changing only two input data points.

The availability of reliable process correlations cost data, and the development of realistic optimum search techniques are the two most important obstacles to the early completion of a useful ultimate program. The small computer user is presently faced with a need to store large blocks of empirical data for sea water processing, in the present state of the art, which makes it necessary to consider off line storage or other difficult and/or expensive means of expanding computer capacity. Correlations now developed for heat transfer coefficients, such as those in use by Houston Research Institute (Reference 2) require complex subroutines and are thus better suited to large computers.

The author, through the analysis of operational data from the Freeport Plant, does not consider any of the present correlations of heat transfer or thermodynamic data to be sufficiently reliable to warrant programming equations instead of storing tabular experience data. Thus, the capacity of the IBM 1620 presents another restraint to further development towards the ultimate program.

Cost data, now being acquired in volume, must be carefully related to process and geometry. The geometry section output must be indexed or coded for capital costing. Progress is being made in these areas. It is expected that the intermediate program will be extended to the cost section by the end of Calendar Year 1966. Optimization search techniques are also being developed to guide the manual manipulation of input data and thus provide a degree of optimization which is consistent with the state of the art.

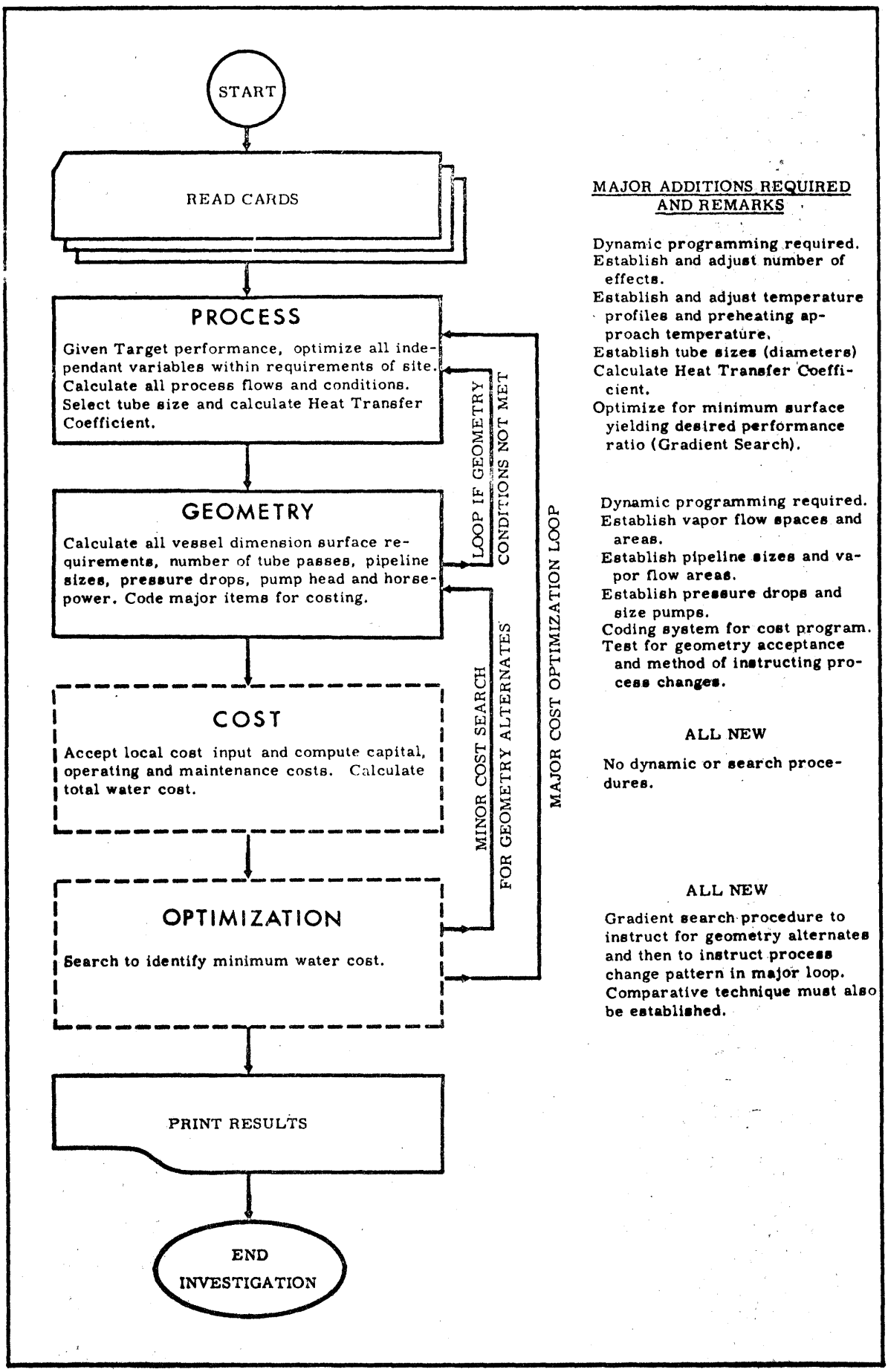


FIGURE 4. GENERALIZED BLOCK DIAGRAM, ULTIMATE DESIGN PROGRAM

VI. ACKNOWLEDGEMENTS

The author gratefully acknowledges the continuing support and assistance of B. P. Duncan in developing this process program. Furthermore, this paper would not have been possible without the support and encouragement of C. G. Rogers, C. F. Yuenger, R. W. Akerlow; and the Stearns-Roger Corporation. Much of the basic data now in use has been derived from the operating records of the Office of Saline Water Plant at Freeport, Texas.

APPENDIX

References

1. Kays, D. D., "Desalting Sea Water in the Multiple Effect-Long Tube Vertical Evaporator", First International Symposium on Water Desalination, Washington D. C., October 3 - 9, 1965.

2. Houston Research Institute, Inc., "The Effect of Tube Geometry and Maldistribution on Heat Transfer in LTV Evaporators of the Freeport Demonstration Plant Type", a portion of the report by Dow Chemical Co., Texas Div., "An Engineering Evaluation of the Long Tube Vertical Falling Film Distillation Process", OSW Contract No. 14-01-0001-397, July 1964.

Symbol List

n	-	An effect of evaporation train containing N effects, where n = 1, 2, 3, . . . N.
W	-	Mass rate of flow per unit time.
W'p	-	Mass rate of steam flow per unit time into an effect.
Wp	-	Mass rate of steam flow per unit time out of an effect.
W'b	-	Mass rate of brine or sea water flow per unit time into an effect.
Wb	-	Mass rate of brine or sea water flow per unit time out of an effect.
W'_{fv}	-	Mass rate of steam flow per unit time out of a condensate flash chamber.
W'c	-	The mass rate of condensed water flow into a flash chamber.
FT	-	Flash chamber for hot condensate.
HXRJ	-	Heat Rejection Condenser
HX30n	-	The exchanger preheating sea water feed in effect n.
HX200	-	The product cooler-feed preheater.
W_{cc}	-	Cold condenser cooling water.
W_{wc}	-	Warm condenser cooling water.

NOTE: All other symbols used in this paper are defined in the legend appearing in Figure 3.

IBM DATA FORMAT - DESALT

SHEET NO. ____ OF ____

PUNCH → STEARNS ROGER CORPORATION ME-FF DESALTING PROGRAM- D8

→ BY CUST.

→ CASE DATE

NO. OF EFFECTS ↓	SFR	TARGET E.R.	STEAM INLET TEMP.	STEAM OUTLET TEMP.	SWF INLET TEMP.	HXRJL	RST	DTRJ	1. HSR x 10^3 OR PR x 10^3
ORANGE CARD →									

→

EFF. NO. 2. ↓	BPR	DTV	DTSWF	VENT LOSS	QL - HEAT LOSS	BFR	UH- . 3. PREHEATER	UE- 4. EVAPORATOR	DTP. TEMP. RISE
→									

NOTES:

1. DO NOT ENTER COMMAS. ENTER ONE ONLY, HSR OR PR, IN THOUSANDS.
2. AN N+1 CARD MUST BE ENTERED AS LAST CARD WITH AN EFFECT NO. OF N+1.
3. ENTER UH-200 ON N+1 CARD.
4. ENTER UE-HRJ ON N+1 CARD.
5. ANY DECIMAL POINT CAN BE OVERRIDDEN.
6. ZERO VALUES MAY BE LEFT BLANK.

64

EXPERIENCES WITH FLBSPS AND STOVE

or They Won't Approve a Faster Computer, What Now?

By K. W. Jones, Department of Highways, State of Colorado

The primary purpose of this paper is to pass on three table handling techniques used at our installation to speed up existing programs as follows:

1. Optimum instead of Logical Ordering
2. Random instead of Sequential Storing and Searching.
3. Pocket instead of Whole Array Sorting

These techniques were utilized in two programs - FLBSPS - "Faster, Less Bulky Symbolic Programming System" (1 and 2) written in SPS and - STOVE - "Sampling, Testing and Overall Variance Etc." (3) written in FORTRAN.

TECHNIQUE ONE

Optimum Instead of Logical Ordering

Chart I displays the ordering of the operation codes in the op code table in SPS II. This table is used to check for valid op codes and to branch to the appropriate handling routine for each accepted code. Also displayed on Chart I are the results of a program written to accumulate the frequency of occurence of each code. The frequencies shown are the results of twenty two source decks which were analyzed by this program. It is apparent that the searching time can be reduced by a rearrangement of this table noting particularly TF, TFM, B, DC and DORG and the blocks of unused operations.

Chart II displays the optimized arrangement in FLBSPS and a tested speed gain for the first minute on a symbol table read for a patch assembly. The speed gain shown is based only on the rearrangement of this table.

TECHNIQUE TWO

Random Instead of Sequential Storing and Searching

Chart III is an attempt to demonstrate how the two above methods utilize a table area. The Sequential Method starts at the first slot in the table and continues down through the table examining each slot until an open slot or duplicate is found when storing or until the sought for label is found when searching. This is demonstrated for the 11th label on the left side of the chart by following the arrows. The Random Address Method uses the actual label being stored manipulated in some way to compute the address of some specific slot somewhere in the table where the label is stored or, if searching, found directly. A problem arises when two different labels happen to generate the address of the same slot and in this case we revert to a sequential search from that slot forward as shown by the arrows on the right side of the chart for the 11th entry.

Chart IV details the program steps used in SPS II to store the label table (Pass I) and to search for operand labels after the table has been completed (Pass II) using the sequential method.

Chart V displays the methods and results of a program written to compute a random address from a label using different methods from which we wanted to select the best. This program was run with a 931 label symbol table as the source of labels. The method shown in the leftmost column (Addition) wherein the 12 numeric digit labels are broken into four 3 digit parts which are added together and adjusted to fit the table area is obviously superior in speed and at least equal in the minimum number of duplicate addresses computed and therefore was selected.

Chart VI details the program steps used in FLBSPS to store the label table (Pass I) and to search for operand labels after the table has been completed (Pass II) utilizing the random address method selected.

Chart VII displays actual test results of the difference in Pass I card reading speeds, again using the symbol table for a patch assembly, and shows a maximum speed gain of approximately 11 times faster with 700 labels when using the random address method of storing instead of the sequential method.

TECHNIQUE THREE

Pocket Instead of Whole Array Sorting

The left column of Chart VIII details statements originally written to perform a core sort which, for each argument, sequences up through previously sorted arguments until a greater value is found and then finds the end of the sorted array finally sliding all greater values ahead one slot and storing this argument. Core storage requirements and test times for 200 numbers are shown and were used as a basis for evaluating the following improvements. The right column details the second method tried and although no speed improvement is shown the memor requirement is far less than the other methods tried.

Chart IX shows the methods and results of two additional methods tested where a double length array for sorting is broken into pockets and each argument is sorted into its proper sequence in its appropriate pocket and finally the used portions of the pockets are put together to form the sorted array. Using 10 pockets we find that storage requirements have more than doubled but we have achieved a time requirement of only 0.12 of our original.

Chart X details the statements used in implementing the 10 pocket sort and reviews the four basic steps necessary.

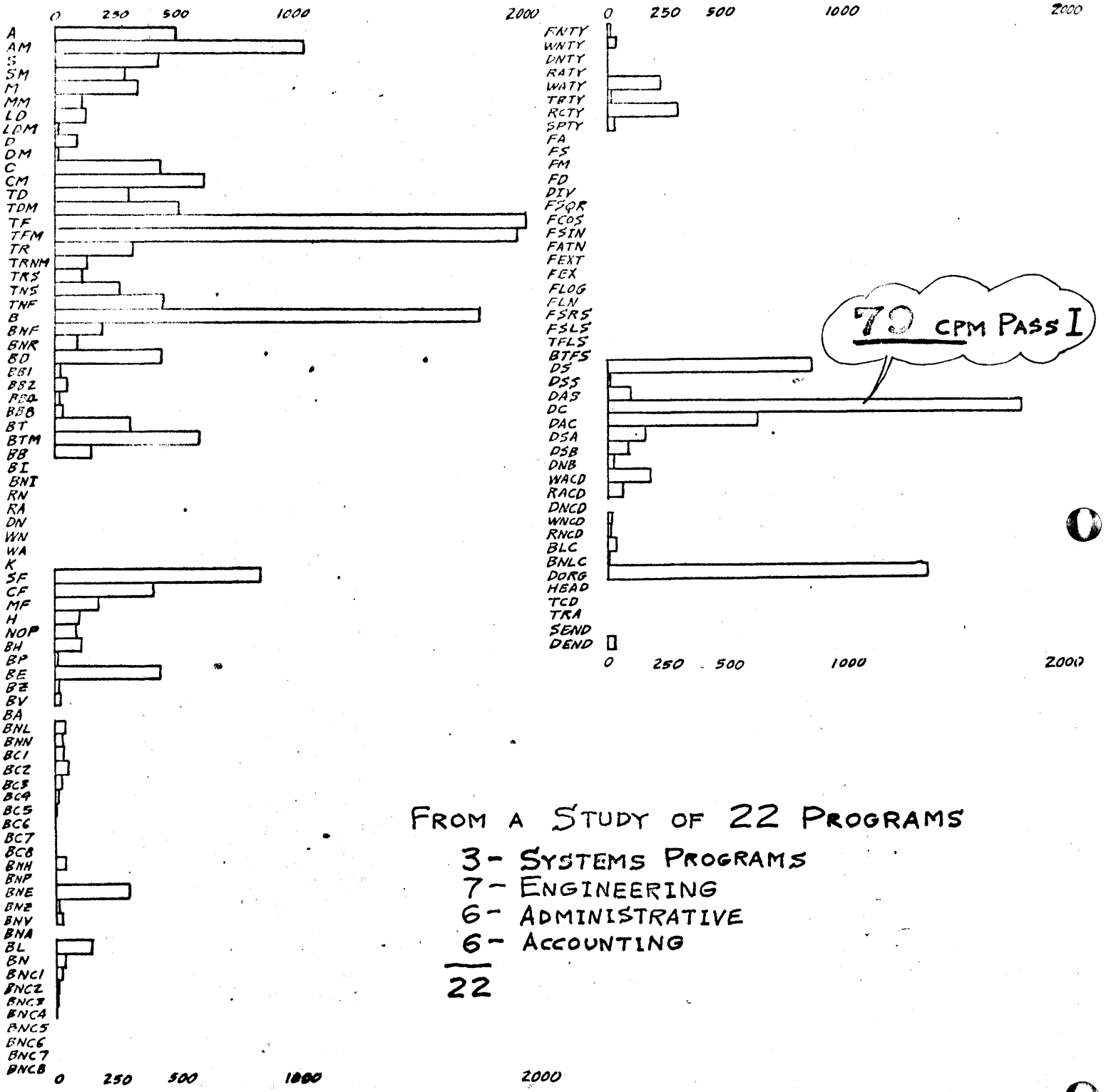
IN SUMMARY

These three techniques are not put forth as the best ways to handle similar problems but rather as significant improvements over the first thoughts a programmer might have.

CHART I

FREQUENCY OF OPERATION OCCURRENCE

LOGICAL ARRANGEMENT



FROM A STUDY OF 22 PROGRAMS

3- SYSTEMS PROGRAMS
7- ENGINEERING
6- ADMINISTRATIVE
6- ACCOUNTING

22

CHART II

FREQUENCY OF OPERATION OCCURRENCE

OPTIMIZED ARRANGEMENT

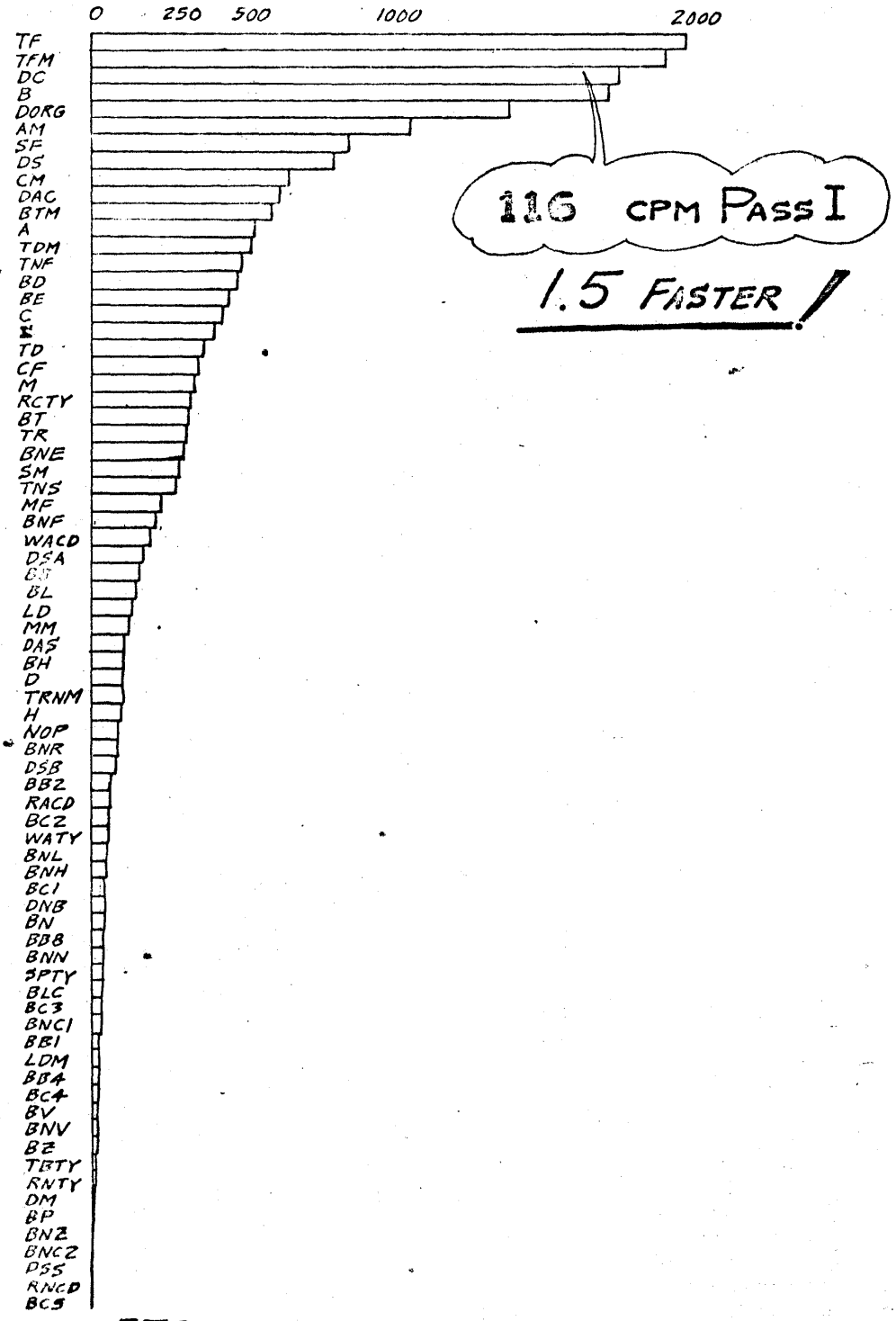
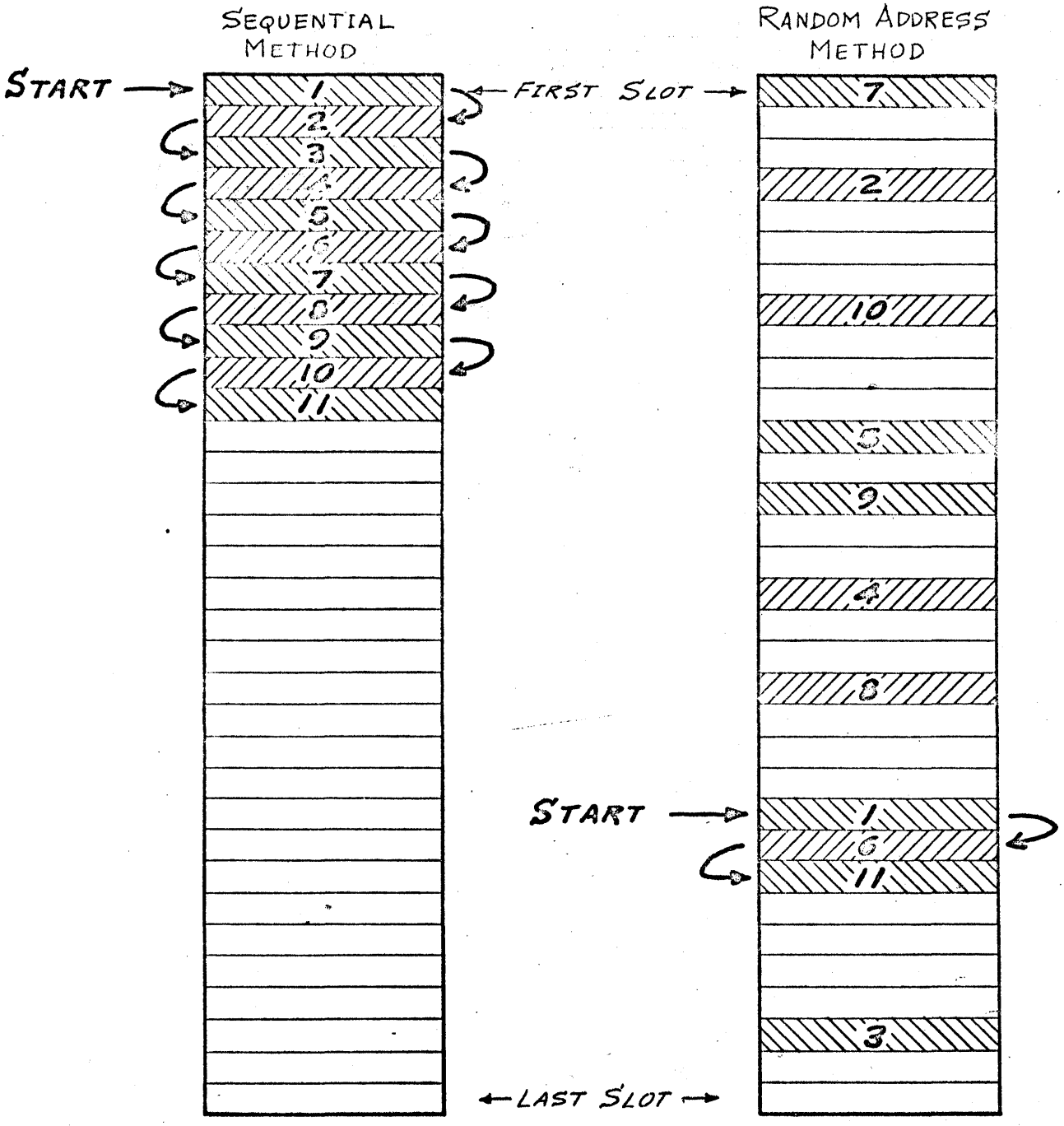


CHART III

TABLE STORING AND SEARCHING



SEQUENTIAL METHOD

LABLE STORING (PASS I)

```
83120 *
83130 *           ROUTINE TO LOAD LABELS INTO SYMBOL TABLE
83140 *
83150 LDLBL  C    CLERER+11,INPUT+10
 ...              ***** COUNT NUMBER OF CHARACTERS *****          ...
84080 TSPEC  SF   LAB-11
 ...              ***** CHECK EACH FOR VALIDITY *****             ...
85010 LDHED  CM   LABCTR,6,10
 ...              ***** ADD HEADING DIGIT IF IS ONE *****         ...
85130 IT     TFM  *+23,SYMTBL          START AT FIRST OF TABLE.
85140        BD   SEIFIN,    ,         IS THIS OPEN SLOT  <------------ *
 ...              ***** STORE LABEL THEN EXIT *****               ...
86110 SEIFIN TF   *+23,IT+23           NO, MOVE ADDR OF LABEL(TABLE) <-+
86120        TD   *+35                 LENGTH AND PEEL IT OFF.         |
86130        TF   *+71,*-1             BUILD ADDRESS FOR CHECK IF      |
86140        AM   *+59,,10             LABEL IS SAME LENGTH.           | *
86150        A    *+47,*-1                                             |
86160        C    LABCTR,SEIFIN+47     IS LABEL SAME LENGTH.           |
86170        BNE  *+36                 <-------------------------------+
86180        C    INPUT3               YES, IS THIS A DUPLICATE. <-----+
86190 *           MULTIPLY DEFINED LABEL                               | #
86200        BE   ER10                 <-------------------------------+
87010        TF   IT+23,*-13           NO, THEN INCREMENT TO NEXT <----+
87020        AM   IT+23,6,10           LABEL AND GO CHECK IT FOR       | *
87030        B    IT+12                A DUPLICATION.  <---------------+
```

0.005 second 1st label
0.4 second 100th label
0.8 second 200th lable
2.1 seconds 500th lable

* 11 INSTRUCTIONS EXECUTED FOR EVERY PREVIOUSLY STORED LABLE UNTIL EMPTY SLOT IS FOUND.

\# 2 ADDITIONAL INSTRUCTIONS EXECUTED FOR EACH LABEL PASSED OF SAME LENGTH.

LABLE SEARCHING (PASS II)

```
35020 *
35030 *           THE SYMBOL TABLE IS SEARCHED FOR EQUIVALENCE
35040 *
35050 LBADD  TFM  *+23,SYMTBL          START AT 1ST OF TABLE.
35060        BD   LBADDS               HAS WHOLE TABLE BEEN LOOKED AT. <-- ϕ
35070 *           UNDEFINED SYMBOL IN OPERAND
35080        BTM  EVALER,50000         YES, THEN TYPE ERROR 5.
35090        DC   1,-,*
35100 LBADDS TF   *+23,LBADD+23        NO, MOVE ADDR OF LABEL(TABLE) <-+
35110        TD   *+35                 LENGTH AND PEEL IT OFF.         |
35120        TF   LABCOM+11,*-1        MOVE ADDR AGAIN FOR 3 DOWN.     |
35130        TFM  *+47,,10             STORE LENGTH THEN               | ϕ
35140        A    *+35,*-1             DOUBLE IT.                      |
35150        A    LABCOM+1,*+23        FINISH BUILDING ADDR LABEL(TABLE). |
35160        CM   COLL-17              IS LENGTH OF LABEL LOOKING FOR  |
35170        BNE  *+36                 SAME AS THIS LABEL(TABLE).      |
35180 LABCOM C    COLL-2               YES, IS THIS THE LABLE WE <-----+
35190        BE   LABOK                ARE LOOKING FOR.  <-------------+ ϕ
35200        AM   LABCOM+11,6,10       NO, INCREMENT TO ADDR OF NEXT <-+
36010        TF   LBADD+23,LABCOM+11   LABEL LENGTH, MOVE IT AND GO    | ϕ
36020        B    LBADD+12             CHECK NEXT LABEL(TA...).  <-----+
```

0.005 second 1st search
0.5 second 100th search
0.9 second 200th search
2.3 seconds 500th search

ϕ 12 INSTRUCTIONS EXECUTED FOR EACH LABLE STORED AHEAD OF THE LABLE BEING SEARCHED FOR.

CHART V

TANTRM METHODS & RESULTS

ADDITION	QUOTIENT	COMBINED (Quot, Add)	REMAINDER
$\overline{X}XX\overline{X}XX\overline{X}XX\overline{X}XX$ (A B C D)	$\overline{X}XXXXXXXXXXX$ (DVR / DD)	$\overline{X}XXXXXXXXXXX$ (LBL)	$\overline{X}XXXXXXXXXXX$ (DD)
$[(A+B+C+D)*L]+S$	$\left[\left(\frac{DD}{DVR}\right)*L\right]+S$	$[(LBL+K)^2*L]+S$	$\frac{DD}{NS}=X+R_{EMAINDER}$ $(R*L)+S$
∴ 6.5	MILLESECONDS PER LABEL CALC. 30.6	30.5	27.4
AVERAGE NUMBER OF DUPLICATE ADDRESSES (931 LABELS - 1000 SLOTS) 1.33	TOO MANY TO COUNT	1.34	

L = Length of table slot
S = Start address of table
K = Constant to prevent bunching
NS = Number of slots in table

72

RANDOM ADDRESS METHOD

LABEL STORING (PASS I)

.008 second 1st label
.010 second average label

```
86200 *
87010 *            ROUTINE TO LOAD LABELS INTO SYMBOL TABLE
87020 *
87030 LDLBL  C     CLERER+11,INPUT+10,,
 ...               ***** COUNT NUMBER OF CHARACTERS *****          ...
87110 ZSPEC  TF    LAB,INPUT+10,,
 ...               ***** CHECK EACH FOR VALIDITY *****            ...
88020 LDHEAD BNF   NOHED,ISHED,,
 ...               ***** ADD HEADING DIGIT IF IS ONE *****        ...
88120 ITS    TR    ARGLOC-4,GRP1-4,,      GO TO SUBROUTINE TO FIND
88130        B     GETLBL,,,              TABLE SLOT AND CHECK FOR
88140        DORG  *-4,,,                 DUPLICATE LABEL.
88150 GRP1   DSA   LAB,STOLBL,ER10,ER9
88160        DC    1,-
```

⊙

LABEL SEARCHING (PASS II)

.012 second 1st label
.015 second average label

```
35050 *
35060 *            THE SYMBOL TABLE IS SEARCHED FOR EQUIVALENCE
35070 *
35080 LBADD  TF    LB,CLERER+11,,         BUILD A 12
35090        TFM   MVLBLB+6,LB-12,,       DIGIT LABEL.
35100        A     MVLBLB+6,COLL-17
35110 MVLBLB TF          ,COLL-2
35120        TR    ARGLOC-4,GRP2-4,,      GO SEE IF
35130        B     GETLBL,,,              LABEL HAS
35140        DORG  *-4,,,                 BEEN DEFINED.
35150 GRP2   DSA   LB,NOTDEF,GOODLB,NOTDEF
35160        DC    1,-
```

Δ

SUBROUTINE COMMOM TO BOTH ABOVE

```
89030 *
89040 *            SUBROUTINE TO SEARCH LABEL TABLE FOR SAME
89050 *            LABEL OR UNUSED SLOT OR NEITHER.
89060 *
89070 ARGLOC DS    5
89080 FREXIT DS    5
89090 DPEXIT DS    5
89100 TILT   DS    5
89110        DS    1
89120 GETLBL TF    WK,ARGLOC,11,          GET LABEL,
89125        CF    WK,,,                  MAKE PLUS,
89130        SF    WK-8,,,                PARTITION
89140        SF    WK-5,,,                INTO 4
89150        A     WK,WK-9,,              SECTIONS THEN
89160        A     WK,WK-6,,              ADD THEM UP
89170        A     WK,WK-3,,              TO COMPUTE
89180        SF    WK-2,,,                SYMBOL TABLE
89190        MM    WK,17,10,              ADDRESS.
89200        AM    99,20000
90010        TF    LOOPBK+11,99,,         SET LOOKED
90020        AM    LOOPBK+11,11,,         THRU WHOLE
90030        TDM   GOBK+1,1,,             TABLE CONTROL.
90040 ISFREE BNR   *+20,99,11,            IS THIS LOCATION FREE.
90050        B     FREXIT,,6,             YES.
90060        DORG  *-3
90070        AM    99,11,,                NO, IS THIS
90080        C     99,ARGLOC,611,         LABEL IDENTICAL.
90090        BE    DPEXIT,,6
90100 GOBK   NOP   LOOPBK,,,             NO, M I WRAPPING AROUND.
90110        CM    99, 9975,,             NO, SHOULD I START
90120        BL    NXTSLT,,,              WRAPPING AROUND.
90130        TFM   99,19994,,             YES.
90140        TDM   GOBK+1,9
90150 NXTSLT AM    99,6,,                 NO, THEN JUST GO
90160        B     ISFREE,,,              ON TO NEXT SLOT.
90170        DORG  *-3
90180 LOOPBK CM    99,     ,,             M I BACK AT
90190        BNE   NXTSLT,,,              STARTING POINT.
90200        B     TILT,,6,               YES.
```

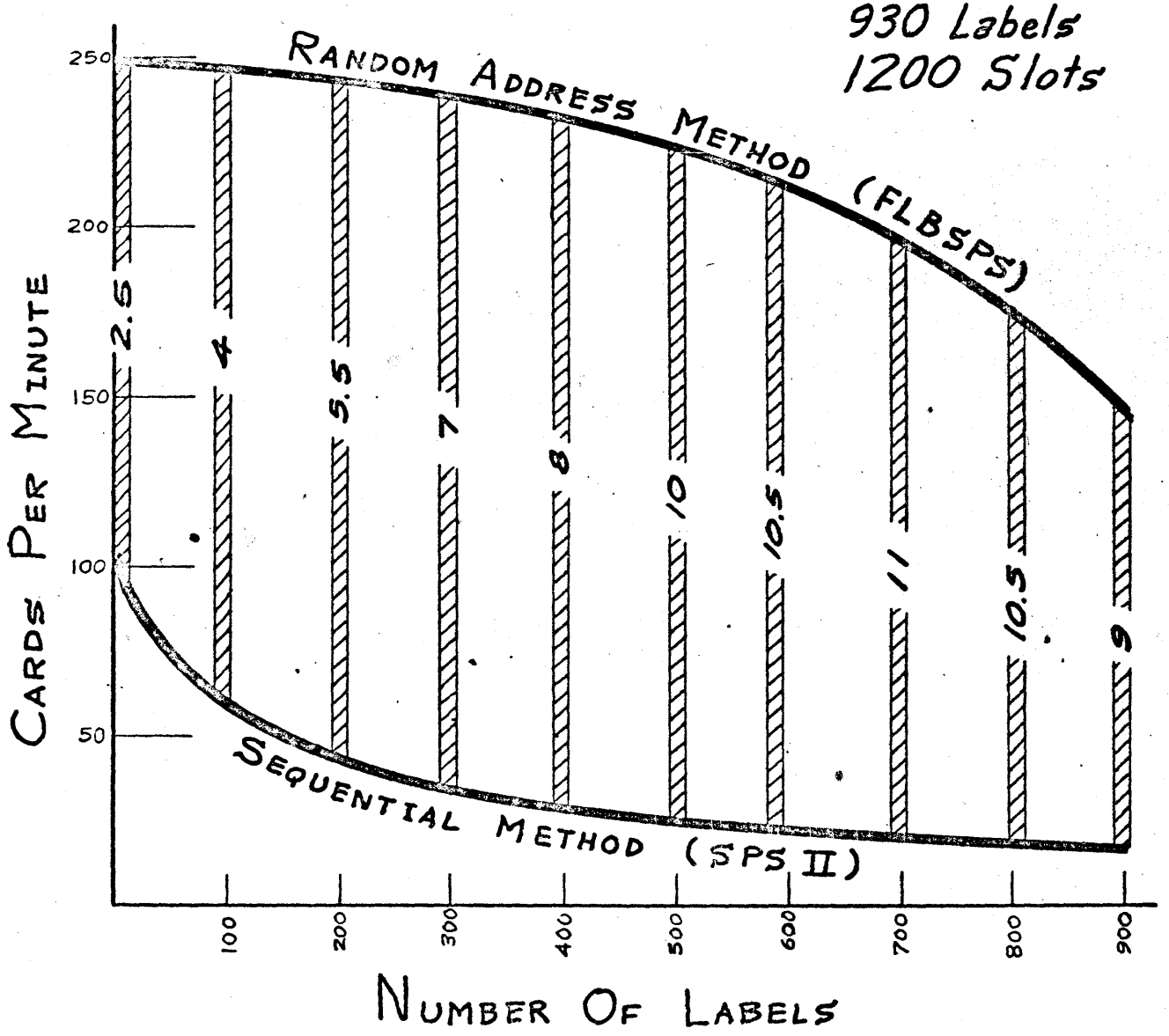
⊙Δ ⊙ Δ

⊙ 17 × 1.3 = ± 22 AVERAGE NUMBER OF INSTRUCTIONS EXECUTED NO MATTER HOW MANY LABELS WERE PREVIOUSLY STORED.

Δ 23 × 1.3 = ± 30 AVERAGE NUMBER OF INSTRUCTIONS EXECUTED NO MATTER WHERE IN THE TABLE THE LABEL BEING LOOKED FOR WAS STORED.

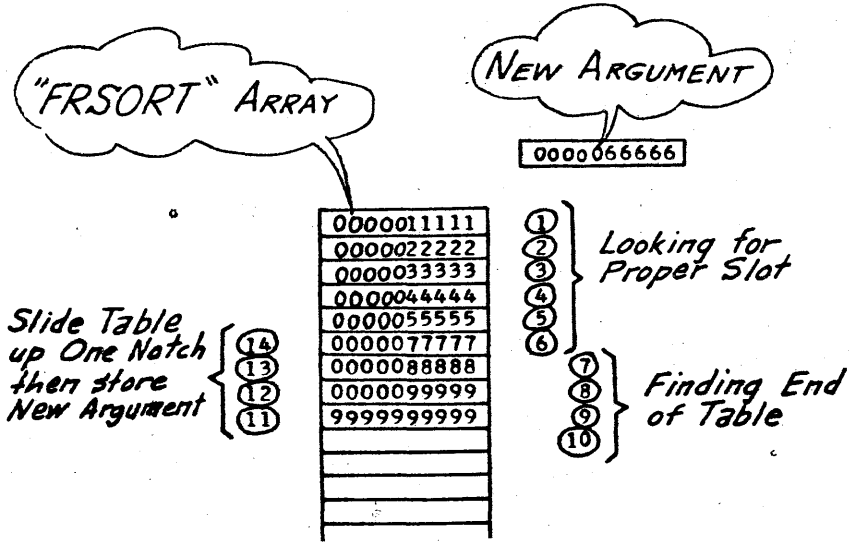
CHART VII

SPEED GAIN ON SYMBOL TABLE READ



STRONG ARM SORT

```
      DIMENSION   STO(400,4),   FRSORT(401)
C
   29 FRSORT(1) = 99999999.
C
      NN = NBRSET(I)              Z SORT THE WHOLE ARRAY
      DO 35 J = 1, NN             Z THAT WAS READ IN.
C
      DO 30 K = 1, 400            Z LOOK FOR GREATER OR
      IF (FRSORT(K) - STO(J,I)) 30,31,31     Z EQUAL.
   30 CONTINUE
C
   31 DO 32 L = K,400             Z NOW FIND TOP OF
      IF (FRSORT(L) - 99999999.) 32,33,94     Z TABLE.
   32 CONTINUE
C
   33 FRSORT(L+1) = FRSORT(L)     Z NEXT SLIDE WHOLE
      IF (L - K) 94,35,34         Z TABLE UP ONE SLOT
   34 L = L - 1                   Z TO OPEN SPACE FOR
      GO TO 33                    Z THIS ENTRY.
C
   35 FRSORT(L) = STO(J,I)        Z NOW STORE THIS AND
C                                    GO DO NEXT IF ANY.
```



REQUIRES ± 970 CORE LOCATIONS PLUS EXTRA SORT BAND	16 MINUTES SORT TIME FOR 200 ITEMS

MIKE'S SORT

```
      DIMENSION   STO(400,4)
C
      MM = NBRSET(I) -1           Z SORT ALL READ IN.
C
      LIMIT = NBRSET(I)           Z GO CLEAR TO TOP 1ST
C                                    TIME WITH LARGEST.
      DO 559 MIKE = 1, MM         Z SORT WHOLE ARRAY.
C
      LIMIT = LIMIT - 1           Z GO 1 LESS FAR THIS
C                                    TIME.
      DO 559 MIKE1 = 1, LIMIT     Z IS THIS VALUE GREATER
      IF (STO(MIKE1,I) - STO(MIKE1+1,I) 559,559,558
C                                    THAN THE NEXT ONE.
  558 HOLD = STO(MIKE1,I)         Z YES, THEN INVERT
      STO(MIKE1,I) = STO(MIKE1+1,I)       Z THESE
      STO(MIKE1+1,I) = HOLD               Z TWO SLOTS.
  559 CONTINUE
```

1.- Move largest value to top of array exchanging inverted pairs on the way.

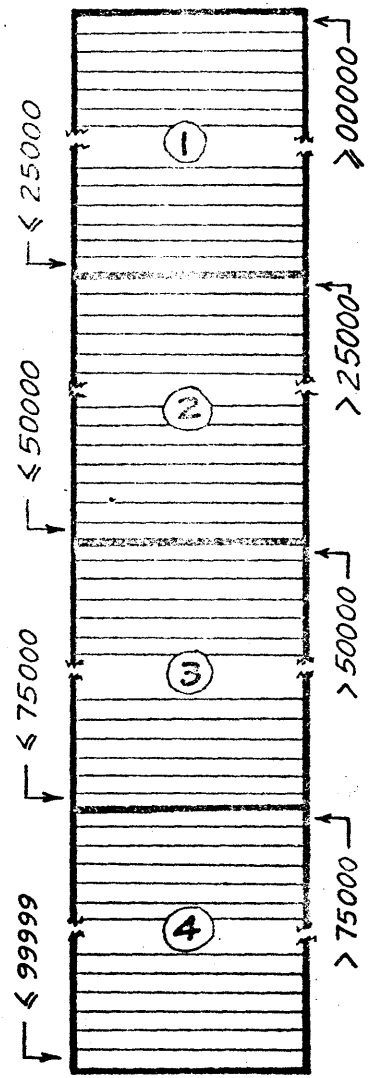
2.- Move next largest value to 1 slot below the top of the array exchanging inverted pairs on the way.

ETC.- down to the smallest value in the first slot.

75

REQUIRES ± 610 CORE LOCATIONS (USES SAME BAND)	17 MINUTES SORT TIME FOR 200 ITEMS

QKSORT



4 POCKETS

I.- Initialize Limits of Pockets from Range of Data.

FOR EACH ARGUMENT:

II.- Determine which Pocket to Put Arg. In.

III.- Sort Arg. into Proper Place In Pocket.

AFTER ALL DONE:

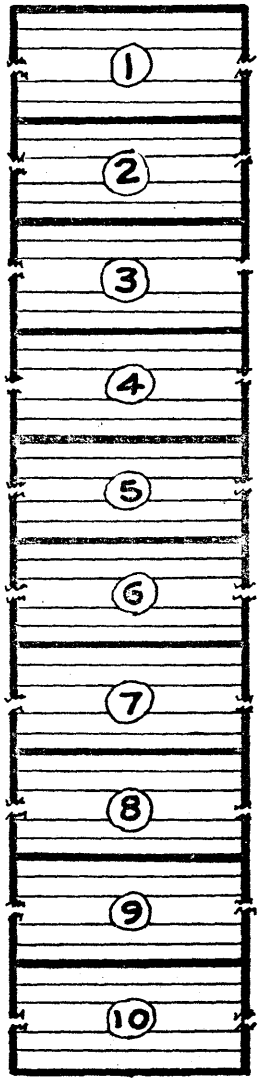
IV. Slide Pockets Together into 1 and 2 for Sorted Array.

0.3

RANGE = 00000 TO 99999

REQUIRES ± 2700 CORE LOCATIONS PLUS EXTRA DOUBLE LENGTH SORT BAND	4 1/2 MINUTES SORT TIME FOR 200 ITEMS

VQSORT



10 POCKETS

The same four steps detailed on the left are used here except a "Normal Distribution" should be used for setting the limits of the pockets under step I.

THIS METHOD REQUIRED ONLY **0.12** OF THE ORIGINAL SORT TIMES

REQUIRES ± 3700 CORE LOCATIONS PLUS EXTRA DOUBLE LENGTH SORT BAND	2 MINUTES SORT TIME FOR 200 ITEMS

CHART

CHART X

VQSORT DETAILED

```
      DIMENSION   STO(400,4),   FS(800)        Z FS SAME AS FRSORT.
C
C     THE DATA IS SORTED INTO TEN POCKETS THEN THESE POCKETS
C     ARE PUT TOGETHER INTO THE FINAL SORTED ARRAY.
C
      DIMENSION   IFAR(10)           Z FARTHEST WAY GONE IN EACH POCKET.
      DO 500 IANYTM = 1, 10
  500 IFAR(IANYTM) = IANYTM * 80 - 79    Z INITIALIZE TO FIRST SLOT EACH ONE.
C
      DIMENSION   BOT(11)       Z CALCULATE MINIMUM VALUE FOR EACH POCKET.
      RNG = SAVMAX(I) - SAVMIN(I)
      BOT( 1) = SAVMIN(I)
      BOT( 2) = SAVMIN(I) + .24*RNG       Z LIMITS REVISED AFTER TEST DUMP.
      BOT( 3) = SAVMIN(I) + .35*RNG
      BOT( 4) = SAVMIN(I) + .40*RNG
      BOT( 5) = SAVMIN(I) + .45*RNG
      BOT( 6) = SAVMIN(I) + .50*RNG
      BOT( 7) = SAVMIN(I) + .55*RNG
      BOT( 8) = SAVMIN(I) + .60*RNG
      BOT( 9) = SAVMIN(I) + .65*RNG
      BOT(10) = SAVMIN(I) + .76*RNG
      BOT(11) = SAVMIN(I) + RNG + ((SAVMIN(I)+RNG) * .1)
C
      IXEND = NBRSET(I)          Z SORT THE WHOLE
      DO 520 IX = 1, IXEND       Z ARRAY READ IN.
      ARG = STO(IX,1)
C
      IPOC = 6                   Z DETERMINE WHICH POCKET THIS GOES IN.
      IF (ARG - BOT(IPOC)) 501,504,504         Z (START IN MIDDLE)
  501 IPOC = IPOC - 1
      IF (ARG - BOT(IPOC)) 501,510,510
  504 IPOC = IPOC + 1
      IF (ARG - BOT(IPOC)) 509,504,504
  509 IPOC = IPOC -1
C     LOOK THRU THIS POCKET AND PLACE IN PROPER POSITION IN ORDER MAKING
C     ALLOWENCE FOR OVERFLOW INTO NEXT POCKET IF THIS ONE IS FULL.
  510 IBEG = IPOC * 80 - 79              Z SET BEGINNING OF LOOK.
      IENDPC = IPOC * 80
      IFR2GO = IFAR(IPOC)                Z SET LAST SLOT PREVIOUSLY USED
      IFAR(IPOC) = IFAR(IPOC) + 1        Z AND INCREMENT FOR NEXT TIME.
      DO 514 IPO = IBEG, IENDPC
C
      IF(IPO - IFR2GO) 512,519,94        Z AM I AT PREVIOUS LIMIT.
  512 IF(ARG - FS(IPO)) 516,516,514      Z NO, IS THIS ARG GREATER THAN SLOT.
  514 CONTINUE                           Z NO, GO UP TO NEXT SLOT IF ANY LEFT.
      IF (IPOC - 10) 515,96,96           Z NONE LEFT THIS POCKET SO GO TO
  515 IPOC = IPOC + 1                    Z NEXT ONE UNLESS IT IS THE LAST ONE.
      GO TO 510
C     NEXT SLOT VALUE IS GREATER THAN ARGUMENT SO SLIDE REST
C     OF POCKET FORWARD LEAVING A SPACE FOR THIS ARGUMENT.
  516 IF (IFR2GO - 801) 617,96,94            Z FIRST IS STORAGE EXCEEDED.
  617 IF (IFR2GO - (IPOC*80+1)) 517,717,717  Z NO, THED DID POCKET OVERFLOW.
  717 IPOC = IPOC + 1                        ZeYES, THEN GO UP TO END OF
      IFR2GO = IFAR(IPOC)                    Z NEXT POCKET BEFORE BEGINNING
      IFAR(IPOC) = IFAR(IPOC) + 1            Z SLIDE.
      GO TO 516
C
  517 FS(IFR2GO) = FS(IFR2GO-1)          Z SLIDE PORTION OF POCKET UP
      IF (IFR2GO-1 - IPO) 94,519,518     Z MAKING ROOM FOR THIS ARGUMENT.
  518 IFR2GO = IFR2GO - 1
      GO TO 517
  519 FS(IPO) = ARG                      Z STO THIS ARGUMENT
  520 CONTINUE                           Z AND GO GET NEXT ONE.
C
C     PUT ALL POCKETS TOGETHER IN THE LOW END OF THE *FS* ARRAY
C
      ITO = IFAR(1)
      DO 539 IPMV = 2, 10                Z MOVE POCKETS 2 THRU 10
      IFROM = IPMV * 80 - 79
C
      IF (ITO - IFROM) 534,531,531       Z DID THIS POCKET FILL UP OR OVERFLOW
  531 ITO = IFAR(IPMV)                   Z YES, SKIP TO NEXT ONE THEN.
      GO TO 539
  534 IF (IFAR(IPMV) - IFROM) 94,539,535       Z NO, WAS THIS POCKET EVEN USED.
  535 FS(ITO) = FS(IFROM)                      Z YES, SLIDE THIS POCKET DOWN.
      ITO = ITO + 1
      IFROM = IFROM + 1
      IF (IFAR(IPMV) - IFROM) 94,539,538     Z R WE PAST LAST SLOT USED.
  538 IF ((IPMV*80+1) - IFROM) 94,539,535    Z NO, R WE PAST LAST SLOT TABLE.
  539 CONTINUE
```

1. INITIALIZE

Set location used thus far in each pocket to "None".

Compute and set Minimum Value to be placed in each pocket.

2. WHICH POCKET?

Pick up next argument.

Determine which pocket this argument should be sorted into starting at center for least tries.

3. SORT INTO POCKET

Determine where in pocket/s this argument belongs

(Similar to the "Strong Arm" Method)

Slide greater values in pocket/s forward one slot leaving a hole for this argument.

Store this argument and go get next one.

4. SLIDE POCKETS TOGETHER INTO ONE SORTED ARRAY

STUDENT SCHEDULING ON THE 1620

by

George L. Crumley
The Citadel
The Military College of South Carolina
Charleston, South Carolina

Presented at
The Western Region Meeting of COMMON
Denver, Colorado
July 6, 7, 8, 1966

At the beginning of the summer vacation last year, we began thinking seriously about a computer registration program for The Citadel. This was undertaken strictly as an academic problem, to see whether or not we could do it. There was no pressure, or even approval at the beginning, from the college administration.

We began by looking at a couple of programs used by other colleges. After considering traditions and certain established ways of doing things at The Citadel, we decided that it would be easier to write our own program than to try to adapt some existing program to our needs. We tried to arrange our disc files and course numbering systems to agree with existing systems and procedures, so that the faculty and administration would not have to accept a complete revision of all their habits. We decided also, near the beginning, that conditions at The Citadel were such that we did not need to place any particular emphasis on speed of the program. Dealing with approximately 2000 cadets, we were not under any great pressure from the point of view of time limitations. Therefore, whenever there was any question of doing a job thoroughly or doing it quickly, we elected to do it thoroughly.

We have a Model 1 1620 computer with 20K memory, one 1311 disc drive, and a 1622 Card Reader-Punch. We also have a 407 accounting machine which we use for printing card output.

We were able to devote an entire disc pack to the scheduling program, and began by loading Monitor I with the SPS assembler and the Disk Utility Program.

The scheduling system consists of a number of programs stored on the disc, all operating under the control of a supervisor called EXEC. It and the Monitor input/output routines remain in core at all times. The input/output routines are necessary because we have used the GET and PUT statements for disc operations. All programs in the

scheduling system are called by use of EXEC control cards. Input can be stacked, except in a few special cases, where the last card indicator is used.

The master schedule is stored in two forms in different places on the disc. The full schedule for every section of every course is stored as a two-sector record. This record is used primarily for printing the cadet schedules and the master schedule. A short form of the section record--giving only the section identification, its meeting times in coded form, and the number of seats available and already assigned--is stored as a one-sector record. This short record is used in the scheduling program itself. Additionally, we have provided a course directory so that the record of any course can be located without a complete search.

Each cadet has a two-sector record showing his identification number, name, his academic major, his class, and his advisor's code number. This is followed by a list of the sections in which he is currently enrolled. There is room for him to be registered for 12 courses, which is certainly more than the Registrar would ever allow him to take in any one semester. Each course that he is taking is represented by a seven-digit entry. Finally, there is an alphameric date which shows the last time that this schedule was changed and a two-digit number which starts at zero with the cadet's original registration and is increased by one every time a change is made in his registration.

The directory that we have provided for the cadet records serves two purposes. First, when we want to find the record of a particular cadet, we can locate it without searching all 2000 records. Second, because we know at all times who is registered, we cannot register the same cadet twice. If we attempt to do this, the act of making a new entry in the directory gives us error messages. This directory is kept in order by cadet number so that we can produce various outputs alphabetically by going through the entire directory.

There is a short faculty directory so that instructor or advisor codes can be translated into alphameric form for printing on cadet schedules, the master schedule, or other output.

To begin each semester's registration, department heads are asked to send the master schedule for their departments to the proper administrative office. There the schedules are completed and coordinated, and are finally punched on cards. By loading the master schedule in the computer, the cadet directories from the last semester are cleared, and both the long and short forms of the section records, as well as the course directory, are stored on the disc.

By using DUMPSK [Dump Schedules], we are able to retrieve the full form of the schedule which was loaded, as well as the name of the instructor of each section, who was identified by code number. These cards are used to cut mimeograph stencils on the 407 so that the full master schedule can be reproduced. Using SHSKD [Short Schedule], we can obtain cards for cutting the stencils for the List of Courses. We have found that both the long and short forms of the Master Schedule have their uses. The average student in enrolling is interested merely in getting the code numbers for the courses for which he wants to register--the short version of the schedule is best for his purposes.

We can pre-punch Course Request Cards for all students who are expected to register the following semester, and these cards, together with the master schedules, are sent to each cadet's faculty advisor.

Each cadet, in conference with his advisor, fills out a Course Request Card, using a five-digit number to identify each course that he wants to take. When these cards are received by the Computer Center, they are key punched, and first run through a program called COUNTM. This program makes a tally of the number of requests for each course and gives us a typewriter message every time a cadet requests a nonexistent course. It also punches a blank Course Request Card for those cadets who ask for nonexistent courses. The output from COUNTM gives us the number of seats available for each course, the number of requests for each course, and then the number of seats still available, or the number of seats short. We examine the original cards of those cadets who asked for nonexistent courses and find that some of these were caused by key-punching errors. Usually we find that the error was made by the cadet. In some cases, we can assume what he probably meant, and we make out a new Course Request Card for him. In other cases, we have no idea what he meant, so we merely delete his request in the corrected Course Request Card. Generally we run COUNTM again after making corrections. The information concerning the number of seats over or short is sent to department heads, who may wish to make changes in the master schedule as a result.

The next step is to sort the Course Request Cards by class, putting seniors first, juniors next, and so on, and then run them under SKEDCA [Schedule Cadets], our primary scheduling program. SKEDCA sets up an array of available sections from the courses requested. First, it ranks the courses requested so that the course with the fewest number of sections open receives the highest priority and the course with the greatest number of sections open receives the lowest priority. After this ranking, the sections within each course are ranked so that the section with the greatest number of seats available will be examined first, and that with the fewest number of seats available will be tried last. After setting up the array, the computer begins to examine every possible combination of sections to find a schedule without conflicts. As soon as such a combination is found, the cadet record is placed on the disc, showing the courses and sections for which he is registered, and the section records are adjusted accordingly, the number of seats

assigned being increased by one, and the number of seats available being decreased by one. Placing the revised section records on the disc completes this cadet's registration. His schedule in printed form can be obtained at any later time.

If all possible combinations of the sections available are tried without a nonconflicting schedule resulting, the computer then types a conflict message. Because of the way our program is written, we feel confident that when our computer declares a conflict to exist, no one on the faculty can produce a schedule by hand that will work. We know of programs used in some colleges that will try 300-500 or some other fixed number of combinations, and then declare a conflict. In a large college with a limited time for scheduling, we can see reasons for following such a procedure--but not in our case. SKEDCA can schedule 2000 cadets in about eight hours, or one cadet approximately every fifteen seconds.

When the initial enrollments have been completed under SKEDCA, we can produce a summary of the enrollment statistics and we can use OUTPUT to produce cadet schedules. These enrollment statistics may give department heads reason to change their schedules either by adding or deleting certain sections. We have a program called ADDSEC [Add a Section] which can be used to change the master schedule.

When the first schedules are received by the cadets, inevitably there will be, for various reasons, required changes. We have two programs available to make changes. The first one, called DROPAD, uses a modified form of SKEDCA to completely revise the cadet's schedule. He is first dropped from every section in which he is enrolled, and then a list of courses requested is built up, first using the courses just dropped with the first Add request. We know that there is room for him in all courses except the new one, because he was in them a few micro-seconds ago. The computer goes through the same process it did under SKEDCA in an attempt to find a schedule with the first additional course. If it is successful in this attempt the course just added becomes a permanent part of his registration, and the second course to be added is tried. If no schedule is found with the last course added, then that course is deleted, and the cadet is given a schedule without it. This procedure is followed for each course to be added.

We have found that the procedure used by SKEDCA usually gives enrollments which are well balanced among the several sections of a given course. In those few cases where the distribution of cadets is noticeably uneven, it seems that the fault lies with the master schedule. The sections have not been offered at the times when they are needed.

We have also found that there is a tendency for cadets to be quite optimistic in any pre-registration. Very few of them predict that they will fail any course. Therefore when the grades come out at

the end of the semester, there is a general tendency for most changes to be from higher to lower numbered courses. This one-way shifting keeps the lower numbered courses well balanced, but sometimes leaves imbalances in those courses from which there was a relatively large exodus.

We can rebalance such sections by running blank Drop-Add Cards for the cadets in those courses.

This DROPAD procedure can be used until and on Registration Day. After Registration Day, it is not desirable, of course, to change the cadet's entire schedule unless it is absolutely necessary. We use the second program, called MANUDA [Manual Drop-Add], for changes in schedule after classes have begun.

After Registration Day, we turn control of section enrollments over to the departments concerned. When a cadet wants to drop or add a course, he and his advisor, having determined what section he wants, fill out a Section Change Card, which must be signed by the head of the department concerned. By signing this card, the department head gives us the authority to register the student in the section specified. MANUDA does this, without changing any other part of the cadet's schedule, and regardless of whether or not the section is still open. MANUDA does not even check to determine whether or not a schedule conflict is created by such a change, for we assume that the cadet and his advisor have done this.

We do not check in this instance because the primary concern here is that we want to give complete control to the department heads concerned. We have found rare cases where, by special arrangements, cadets were to be scheduled for courses that appear to conflict.

We have an artificial department, number 75, which offers non-credit courses designed to help give athletes light schedules in the afternoon. We set up schedules for three sections of one of these courses, the first of which has the greatest number of seats available, and whose time schedule takes up most of the late afternoon; the second has a smaller number of seats available and takes up somewhat less afternoon time and the third has even fewer seats available, but does not take up any time at all. By trying to fit these "classes" into the cadet's schedule, SKEDCA will schedule his real classes in the morning hours if it is possible. In the end, however, he may be scheduled for the third section, which keeps us from getting a conflict. In this manner, we attempt to save the athlete's afternoons for practice but we will not sacrifice an enrollment in any academic class in order to do so.

The possibilities for misuse of this device are obvious to anyone who is familiar with the resourcefulness of college students. The course number for this course is not published in the master schedule and is not printed on the schedules of the cadets who are

enrolled in it. The Director of Athletics sends a list of eligible cadets to the Registrar, who then fills out Drop-Add Cards for the cadets involved. To provide greater security, the number of this course is changed from one semester to the next. In the event of a "leak", it could even be changed during an enrollment, if necessary.

At the present time this is the only use we make of our Department 75. Other colleges could use it to schedule a lunch hour, or to free certain hours for outside work.

At the end of Registration Day, we punch Course Cards for every course (except those in Department 75) in which every cadet is enrolled. These cards are sorted by the sorting code, and then are used to print section rosters for the instructors.

After Registration Day there will be a certain number of changes by Section Change Cards. When the Registrar has stopped such changes, we punch a new set of cards, called Grade Cards, which are printed differently but are punched exactly the same as Course Cards. These cards are sorted by sections and are used to provide final section rosters for the instructors. The same cards are kept for use as Grade Cards at mid-semester, and copies of them are made for use as Grade Cards at the end of the semester.

Since the enrollment of every cadet at The Citadel is on the computer disc throughout the semester, we are able to produce rosters of the entire Corps of Cadets for the Registrar's Office, rosters of cadets majoring in their departments for all department heads, and almost any kind of roster for anyone who has a legitimate use for it, at almost any time. These rosters are automatically kept up to date as the result of searching cadet records for desired information. The cadet records are kept up to date by the use of Drop-Add or Delete Cards.

Changes of major are completed by sending a Change of Major Request Card to the computer. The program, MAJORX, looks up the former major and advisor from the cadet record and then substitutes the new information on the cadet record. Then it produces cards for printing notices to be sent to everyone concerned.

Near the end of a semester, when we are ready to begin pre-registration for the following semester, we can obtain pre-punched Course Request Cards.

Since this set of programs has been written as an academic exercise and not as a regular part of our duties, we have been under no particular obligation to provide documentation. We have drawn flow charts for only a few programs where they seemed unavoidable.

We have written a scheduling manual which has been distributed to the faculty at The Citadel and to those attending the COMMON meeting in Denver. The manual was designed to acquaint The Citadel faculty with the fundamental steps in planning and setting up courses and sections for the coming semester, as well as to provide general and specific guidance concerning details of registration and related procedures. It also contains operating procedures for Computer Center personnel. This manual is not printed as a part of the Proceedings, but may be obtained by writing to

George L. Crumley
Box 79, The Citadel
Charleston, S. C. 29409

WHITMAN COLLEGE REGISTRATION AND GRADING SYSTEM

Whitman College is an independent four year liberal arts institution with an enrollment of approximately 1100 students. As part of its commitment to high quality education, Whitman College established in the spring of 1964 a computer center for faculty, student, and administrative use. The original computer installation consisted of an IBM 1620 computer with a 1622 card reader-punch, sorter, keypunch, and a 407 accounting machine. The computer installation has recently acquired a 1311 disk drive, a 1443 on-line printer, a gangpunch device on the original keypunch and an interpreting keypunch.

The first version of the registration and grading system used the original computer configuration. A Class Card and a Student Master File were designed and two programs were written to do most of the desired work. The registration information program printed student information such as his schedule of classes, addresses, data on parents, etc. for the use of other administrative offices. The grade processing program printed grade reports, probation lists, honors lists, rank in class, and fraternity-sorority grades. Other operations such as class lists were done using only the 407. The results of the initial system were encouraging--a large amount of work was being performed satisfactorily on the computer. There were some limitations in the system because the original computer equipment was inadequate. Without the on-line printer it was necessary to punch cards and carry the cards to the 407 for printing. This became a problem when the number of print lines in the registration information report reached 25,000 lines and required the punching of 50,000 cards. Both the registration information program and the grade processing program required the merging of the Student Master File with the Class Cards. This is a cumbersome operation without a collator.

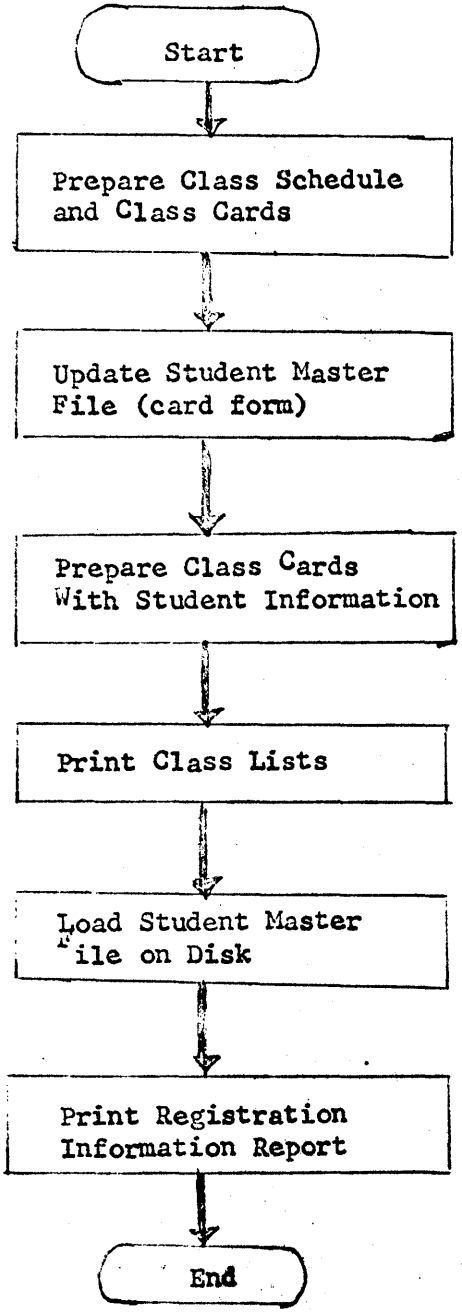
When the Computer Center acquired additional equipment, an improved registration and grading system was made possible. In planning for the improved system it was necessary to decide whether or not the Student Master File should be permanently stored on a disk pack and if so, whether the File should be stored on a monitor pack or be independent of the monitor. Programs could be written which use the disk drive and which are independent of the monitor disk input-output routines, but this would have complicated the programming for the disk file. Therefore, it was decided to use the monitor disk input-output routines, thus requiring storage of the Student Master File on a monitor disk pack.

The Student Master File contains a large amount of alphabetic information. This combined with the two-digit alphabetic representation of the 1620 means that the disk records will be of large size. Presently the disk records of the Student Master File are 16 sectors or 1600 digits of disk memory. Since Whitman College has an enrollment of approximately 1100 students, the disk form of the Student Master File requires 1,760,000 digits of disk memory. In addition there must be memory available for expansion of the disk record.

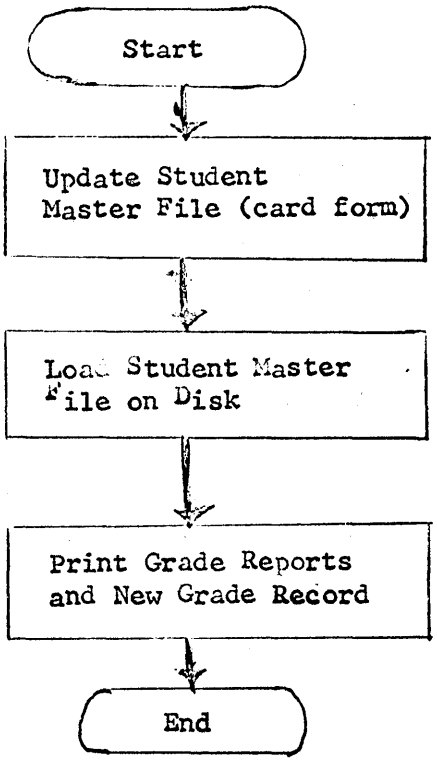
Since there are only 2,000,000 digits of memory on a disk pack and the monitor program requires about 440,000 digits of memory, approximately 1,560,000 digits of disk memory are available for data storage. Therefore, more than one disk pack is required to store the disk form of the Student Master File. Any data file using more than one disk pack on a single drive system cannot be handled readily as a permanent disk stored data file because the programming of maintenance programs for new records would be too complex.

To get around the difficulties of a permanent disk stored file, it was decided to make all corrections, additions, and deletions to a card form of the Student Master File and to load the card form onto a disk pack prior to processing registration information or grades. This eliminates the need for a collator to merge Student Master File and Class Cards for processing. In practice card handling has almost been eliminated. Also selective runs on various sets of class cards can be made with unused disk master records being bypassed.

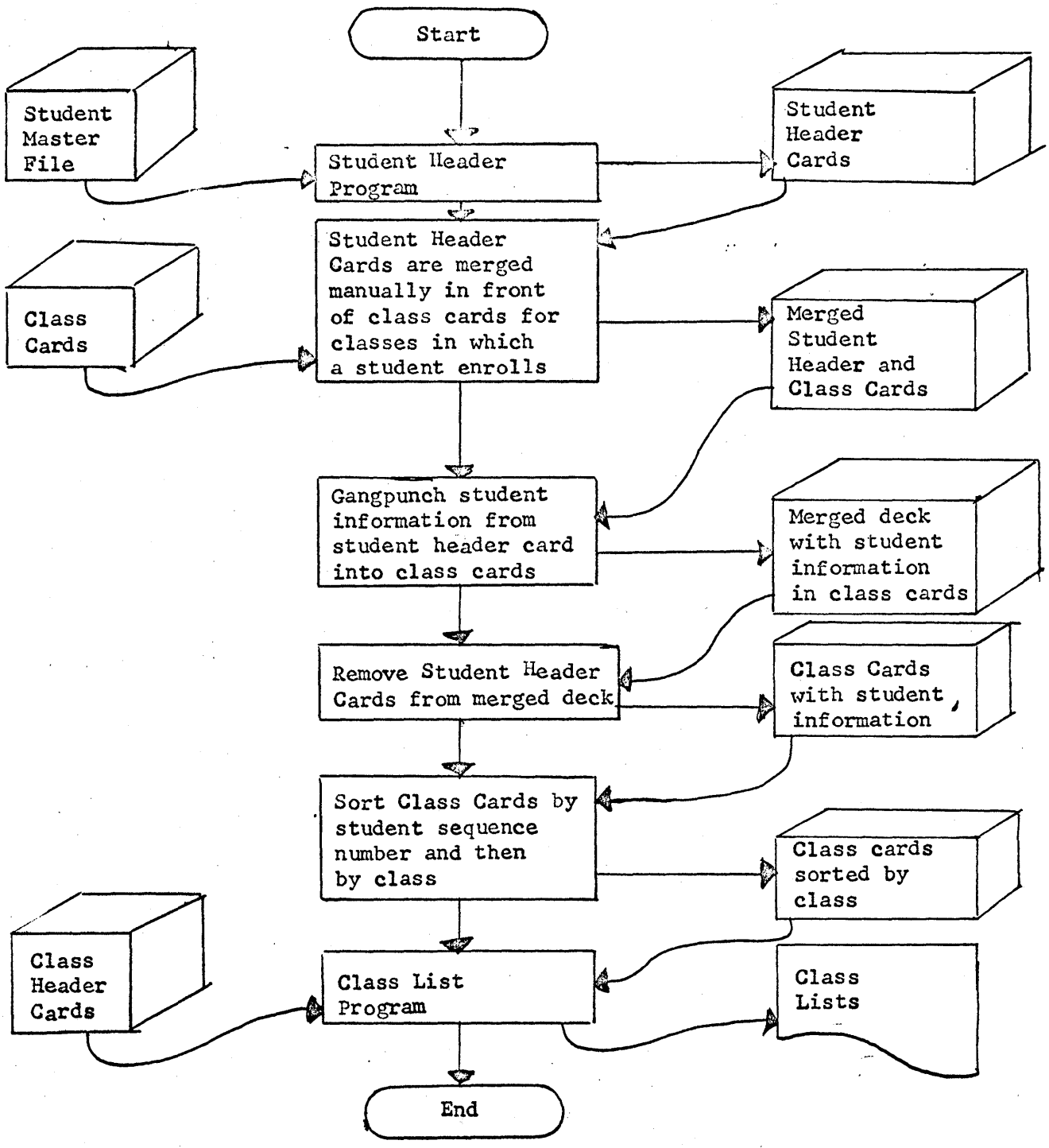
REGISTRATION SYSTEM



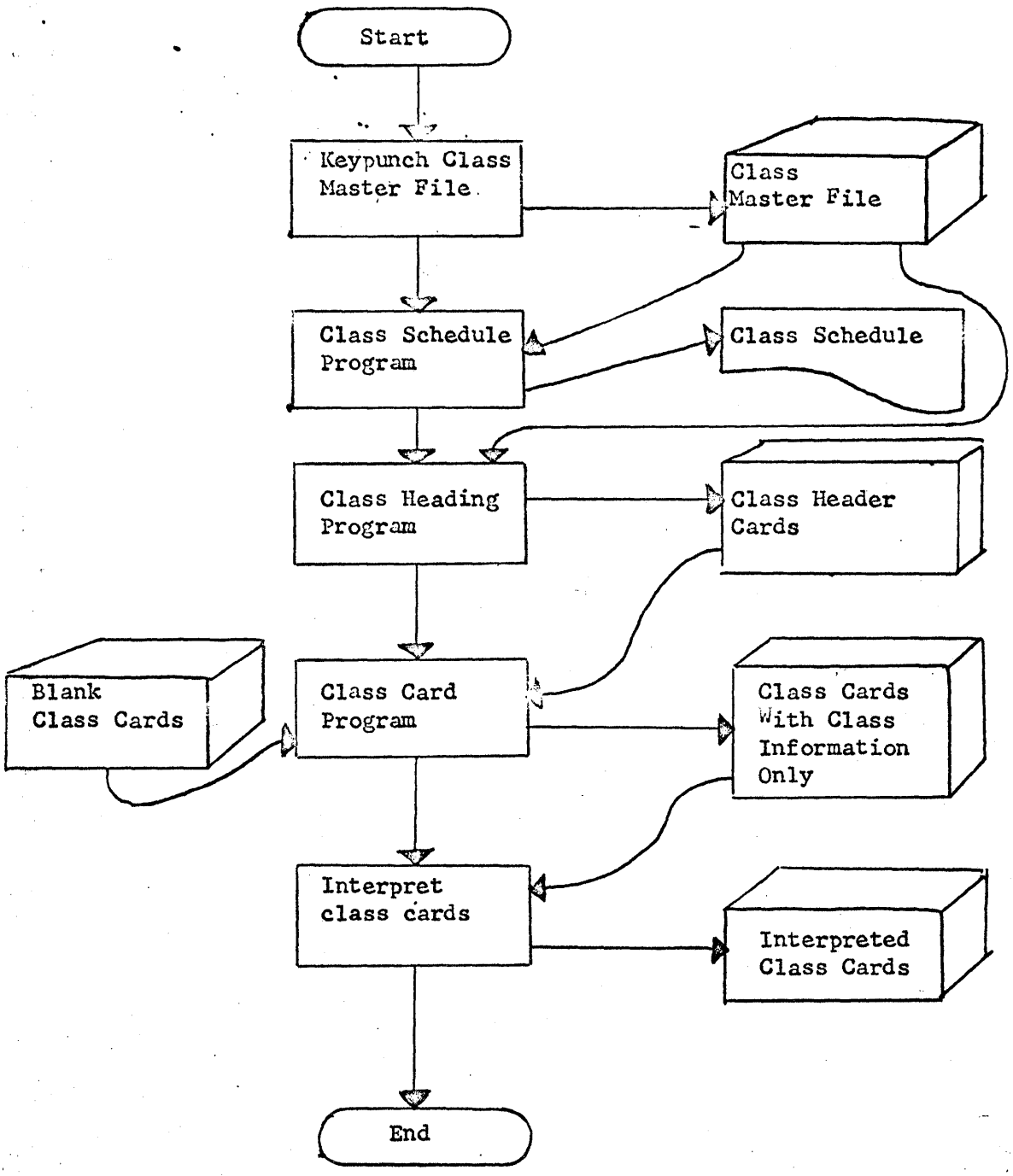
GRADE PROCESSING



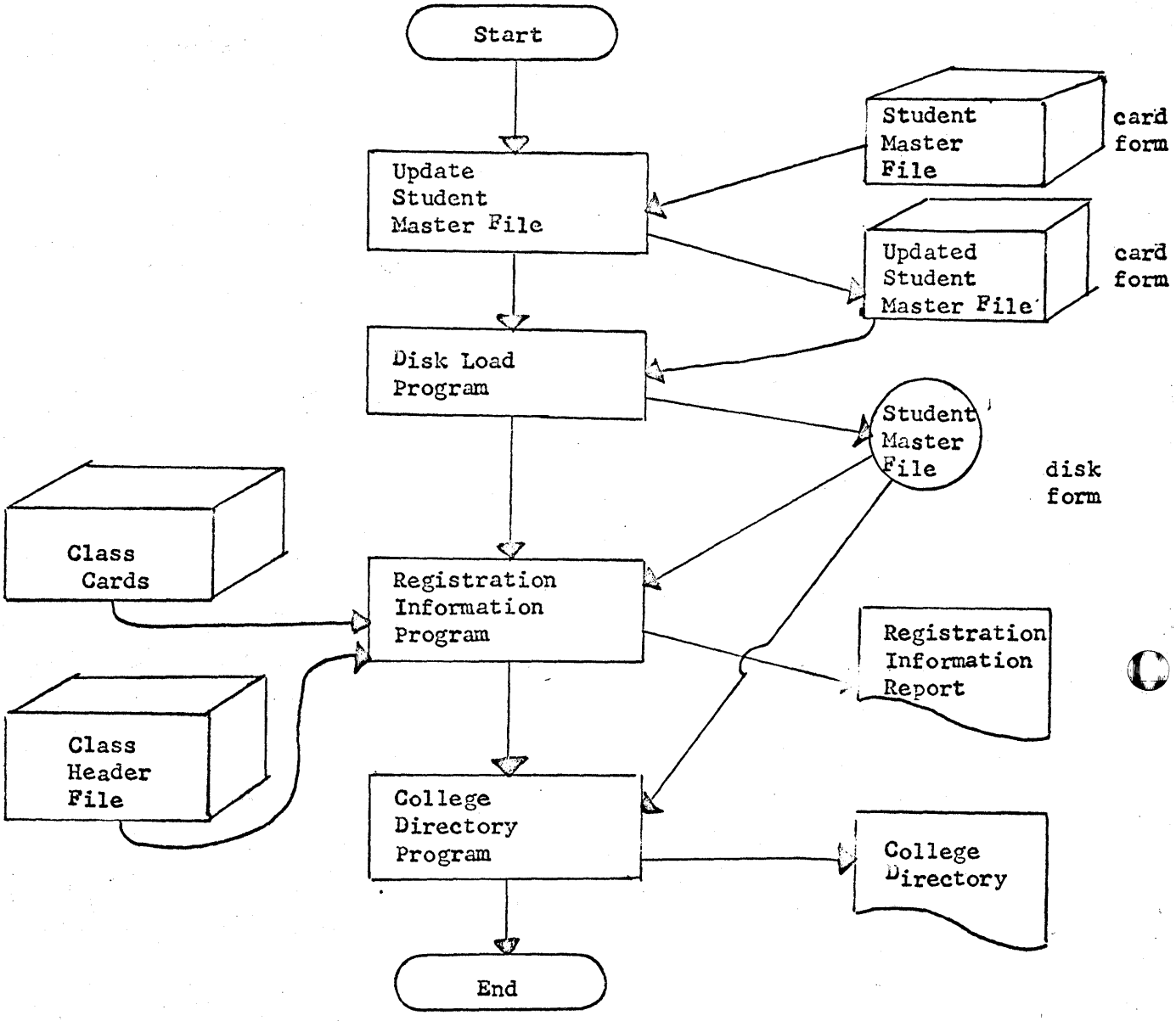
STUDENT REGISTRATION



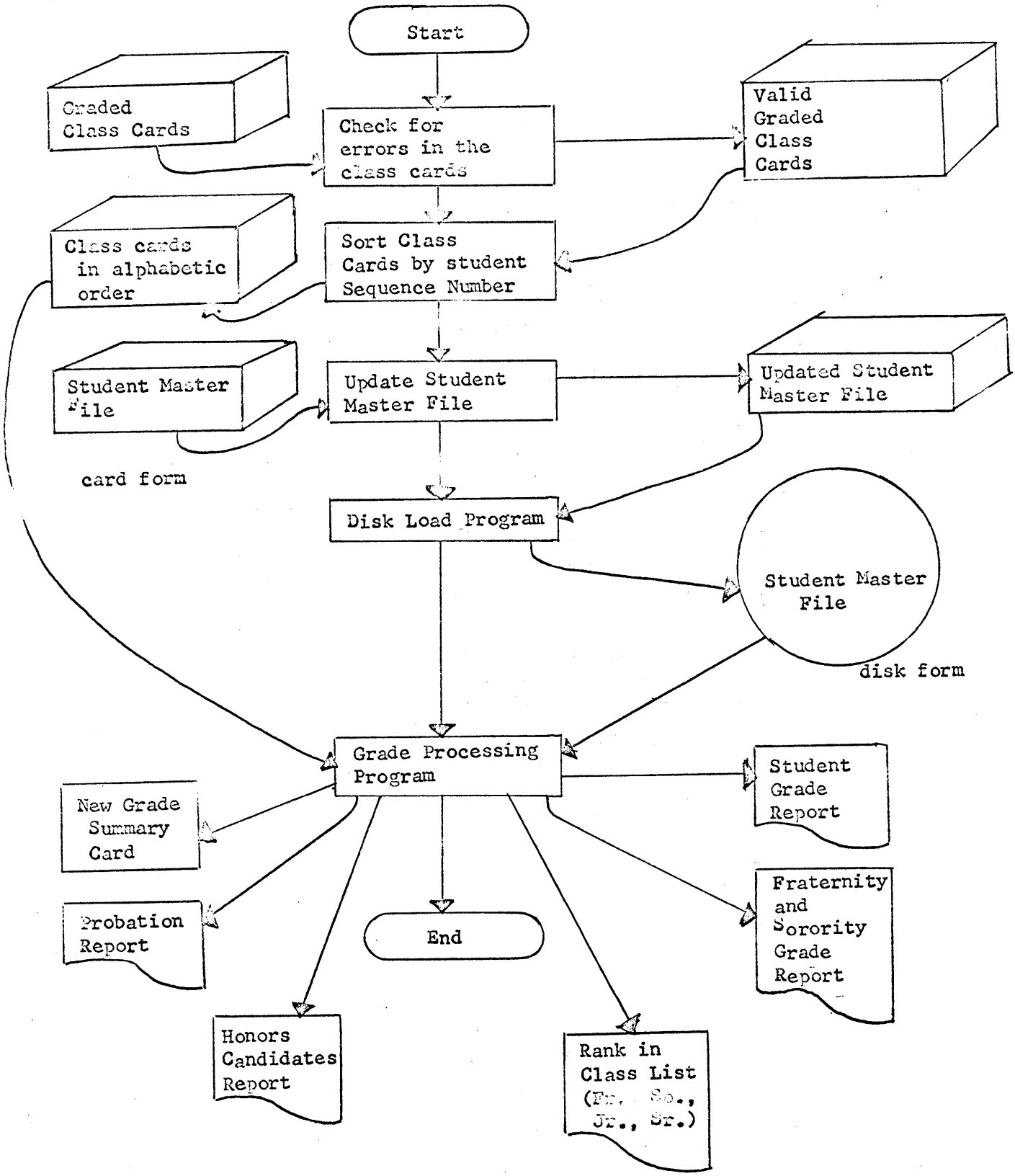
PREPARATION OF CLASS SCHEDULE AND CLASS CARDS



REGISTRATION INFORMATION REPORTS



GRADE PROCESSING



SYSTEM SUMMARY

I. Data Files

a. Class Master File
b. Class Header File
c. Student Master File
   1. [illegible] form
   2. disk form
d. Student Header File

II. Reports

a. Class Lists
b. Registration Information
c. Grade Reports
d. Probation Lists
e. Honors Lists
f. Fraternity and Sorority Grade Lists
g. Rank in Class

III. Documents

a. Class Schedule
b. College Directory

IV. Programs

a. Class Schedule Program
b. Class Heading Program
c. Class Card Program
d. Student Header Program
e. Class List Program
f. Disk Load Program
g. Registration Information Program
h. College Directory Program
i. Grade Processing Program

CLASS MASTER FILE

There are master cards for each class and each laboratory section. The comments card is used for printing of the class schedule and applies to both classes and laboratory sections.

Lecture Master Cards

Card Column	Contents
1-4	Department Code
5-6	1st Semester Course Number
7	1st Semester Footnote Symbol
8-9	2nd Semester Course Number
10	2nd Semester Footnote Symbol
11-29	Course Description
30-32	Hours of Credit
33-36	Course Number and Section
37-47	Time
48-62	Instructor
63-71	Room
76	Special Case Code
77	Time Code
78-79	Department Number
80	"+" first card for each class "-" succeeding lecture cards for each class

Laboratory Master Cards

Card Column	Contents
1-4	Department Code
5-6	1st Semester Course Number
7	1st Semester Footnote Symbol
8-9	2nd Semester Course Number
10	2nd Semester Footnote Symbol
11-29	Course Description
33-36	"Lab" and Section
37-47	Time
48-62	Instructor
63-71	Room
76	Special Case Code
77	Time Code
80	"o"

CLASS MASTER FILE (Cont.)

Comment Cards

Card Column	Contents
1-4	Department Code
5-6	1st Semester Course Number
7	1st Semester Footnote Symbol
8-9	2nd Semester Course Number
10	2nd Semester Footnote Symbol
12-29	Commentary
78-79	Department Number
80	Card Code

CLASS HEADER FILE

Card Column	Contents
1-11	Time
12-20	Room
21-35	Instructor
38-41	Department
42-43	Course Number
44-45	Section
47-48	Department Code
49-67	Course Name
68-69	Hours (scaled (1,1))
77-79	Number of Places in class
80	"o" card code

STUDENT HEADER FILE

Card Column	Contents
1-9	Social Security
10-13	Sequence Number
14-35	Name
36	Sex
37	Classification
80	"o" card code

STUDENT MASTER FILE

NAME CARD NO. 1 (Gray)

1-9	Social Security Number
10-13	Sequence Number
14-35	Name
36	Sex
37-42	Birthdate
43-57	Birthplace
58-63	High School (CEEB Code)
64-67	Date Graduated from High School
68-78	Blank
79	Classification at Entrance (7 indicates freshman with credit)
80	"1" punch

RELATIVES CARD NO. 2 (no Stripe)

Not in all student packets
Essentially free form
Abbreviation followed by name
Different people separated by commas

80	"2" punch

TEST SCORE CARD NO. 3 (green)

1-9	Social Security Number
10-13	Sequence Number
14-17	High School GPA
18-21	High School GPA (Solids)
22-30	Rank in High School Class (Slash in Column 26 must be there for statistical studies)
31-34	Grade prediction
35-38	Date of CEEB Scores
39-41	Verbal Score
42-43	Percentile for Verbal Score
44-46	Mathematical Score
47-48	Percentile for Mathematical Score
49-52	Date of Achievement Exams
53-73	Data for three tests 2 digit Identifier 3 digit Score 2 digit Percentile
74-79	Matriculation Date
80	"3" punch

HOME ADDRESS AT ENTRANCE CARD NO. 4

1-9	Social Security Number
10-13	Sequence Number
14-34	Number and Street*
35-47	City*
48-61	State & Zip (Country)*
62-67	Continent*
68-69	Blank
70-79	Church Preference
80	"4" punch

*Home address at entrance. Left adjusted on cards.

HOME ADDRESS CARD NO. 5

1-9	Social Security Number		
10-13	Sequence Number		
14-34	Number and Street	-21	Left
35-47	City	-13	Adjusted
48-61	State & Zip (Country)	-14	on Cards
62-67	Continent	-6	
68-69	State Code (For geographic distribution)		
70-72	Blank		
73-79	Home Phone (No area code)		
80	"5" punch		

LOCAL ADDRESS CARD NO. 6 (Blue)

1-67	Same Format as Home Address Card
68-72	Blank
73-79	Local Telephone Number
80	"6" punch

PARENTS NAME CARD NO. 7 (Rose)

1-9	Social Security Number
10-13	Sequence Number
14-30	Title
31-52	Name (First, Initial, Last)
53-76	Blank
77	M or F (Mother or Father) if last name of parent is different from last name of student
78	G if Guardian (only if guardian is not father)
79	"1", "2", or "3" depending on order of name and address cards.*
80	"7" punch

**PARENTS ADDRESS CARD NO. 8 (Rose)

1-67	Same Format as Home address card
68-78	Blank
79	"1", "2", or "3" depending or order of name and addres cards.*
80	"8" punch

*There are possibly 3 sets of #7 & #8 cards. They are distinguished by 1,2, 3 in column 79 of the 7 and 8 card.

PARENTS INFORMATION CARD NO. 9 (Salmon)

1-9	Social Security Number
10-13	Sequence Number
14	Father Living (y or n)
15-30	Father's Occupation
31-46	Name of Firm
47	Mother Living (y or n)
48-63	Mother's Occupation
64-79	Name of Firm
80	"9" punch

PERSONAL DATA CARD NO. 10 (Brown)

1-9	Social Security Number
10-13	Sequence Number
14	Marital Status (m or S)
15-29	Name of Spouse
30	Car Operation (Y or N)
31-32	Fraternity
33	Veteran Status
34-37	Degree of Self-support
38-46	Re-entry dates (3 digits each)
47-52	Counselor Code (2 3-digit codes)
53-67	Counselor (s)
68-71	Major (2-digit code) Two majors possible
72-79	Other Colleges attended (CEEB codes)
80	"0" punch

** This card appears only if parents' address is different from home address.

GRADE RECORD CARD NO. 11 (Purple)

1-9	Social Security Number
10-13	Sequence Number
14	Graduation on Schedule 1 - yes 3 - Late 2 - early blank- Did not graduate
15-21	Rank in Class (Punch rank in class at end of each semester, but by-pass incomplete and deferred grades)
22-25	Cumulative GPA to date (By-pass incomplete and deferred grades)
26-29	Cumulative Grade Points (By-pass incomplete and deferred grades)
30-35	Graduate Record Examination Date
36-37	Test Identifier
38-40	Score
41-42	Percentile
43	Classification
44	Number of Semesters in Residence
45-48	Hours to Date (Including Credit)
49-51	Non-graded Hours to Date
52-54	Cumulative Hours of F
55-57	Previous Semester Graded Hours
58-60	Previous Semester Grade Points
61-63	Transfer Hours
64-66	Hours of F in last semester
67-75	Scholastic Cases 1 - Dean's List 2 - Undergraduate Honors 3 - Scholastic Probation 4 - Dropped
76-79	Blank
80	"-" punch

SOCIAL SECURITY NO. | SEQ. NO. | STUDENT NAME | M | F | DEPT. | NO. | SEC | S | COURSE NAME | HRS | G | POINTS | P | YEAR | CC

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

DO NOT WRITE ABOVE THIS LINE

WHITMAN COLLEGE

TO THE STUDENT:

FILL IN THIS PORTION OF THE CARD.

LAST NAME FIRST NAME MIDDLE NAME

CLASSIFICATION: FR____ SO____ JR____ SR____ GR____ SP____

DO NOT FOLD, TEAR, OR DAMAGE THIS CARD.

IBM J74471

0 1 2 3 4 5 6 7 8 9

TO THE INSTRUCTOR:

TO REPORT FINAL GRADE, USE SHARP PENCIL TO PUNCH DESIRED GRADE. IF AN ERROR IS MADE, PUNCH THE CORRECT GRADE AND CIRCLE CORRECTION.

INSTRUCTOR'S INITIALS

- ☐ A
- ☐ B
- ☐ C
- ☐ D
- ☐ F
- ☐ Cr
- ☐ WF
- ☐ WP
- ☐ W
- ☐ IP
- ☐ IF

CLASS CARD

SOCIAL SECURITY NO. | SEQ. NO. | STUDENT NAME | M | F | DEPT. | NO. | SEC | S | COURSE NAME | G | YEAR | CC

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

DO NOT WRITE ABOVE THIS LINE

WHITMAN COLLEGE

TO THE STUDENT:

FILL IN THIS PORTION OF THE CARD

LAST NAME FIRST NAME MIDDLE NAME

CLASSIFICATION: FR____ SO____ JR____ SR____ GR____ SP____

DESIGNATE LECTURE SECTION, IF ANY _____

IBM J74470

0 1 2 3 4 5 6 7 8 9

LABORATORY SECTION CARD

2867

STUDENT'S NAME, HOME ADDRESS & TELEPHONE	DATE	SEX	VETERAN	PARENT'S OR GUARDIAN'S NAME & ADDRESS	BIRTH DATE	BIRTHPLACE
	STUDENT'S LOCAL ADDRESS & TELEPHONE				CHURCH PREFERENCE	SELF-SUPPORT.
					RELATIVES, INCLUDING SPOUSE, IF ANY, WHO ATTENDED WHITMAN AND THEIR RELATIONSHIP TO YOU.	
FRATERNITY	AUTOMOBILE	MARRIED	NAME OF SPOUSE	FATHER LIVING	MOTHER LIVING	
COUNSELOR	MAJOR	CLASSIFICATION	HIGH SCHOOL	DATE GRADUATED		

COURSE	NO.	SEC.	CR.	TIME	ROOM	INSTRUCTOR	OTHER COLLEGES ATTENDED & DATES

CAMPUS ACTIVITIES & HONORS IN COLLEGE

FATHER'S OCCUPATION

NAME OF FIRM

MOTHER'S OCCUPATION

NAME OF FIRM

Registration Information Report

JONES JOHN

WHITMAN COLLEGE
WALLA WALLA,WASH.
SEMESTER GRADE REPORT

HIST 27	ENGLISH HISTORY	3.0	B	9.0
HIST 55	HIST OF FAR EAST	3.0	B	9.0
HIST 71	HISTORY OF RUSSIA	3.0	A	12.0
ART 11	ART APPRECIATION	3.0	B	9.0
GEOL 21	HISTORICAL GEOLOGY	4.0	A	16.0

MR AND MRS ROBERT JONES
4336 PENSHURST CT
SACRAMENTO
CALIF 95825

SEM GRADED HOURS	16.0
SEM GRADE PTS	55.0
SEM GPA	3.437
TOT WHITMAN GR HRS	094.0
CUM GRADE PTS	286.0
CUM GPA	3.042
TOTAL HRS EARNED	128.0

101

DOE JOE

WHITMAN COLLEGE
WALLA WALLA,WASH.
SEMESTER GRADE REPORT

SOC 64	POP - HUMAN ECOLOGY	3.0	B	9.0
HIST 51A	AMER REVOL - CONST	3.0	C	6.0
CHEM 91	PHYSICAL CHEMISTRY	4.0	B	12.0
BIOL 51	VERT EMBRYOLOGY	5.0	B	15.0
CHEM 93	SEMINAR	1.0	C	2.0

MR AND MRS JAMES DOE
7004 SE 20TH
MERCER ISL
WASH98046

SEM GRADED HOURS	16.0
SEM GRADE PTS	44.0
SEM GPA	2.750
TOT WHITMAN GR HRS	122.0
CUM GRADE PTS	320.0
CUM GPA	2.622
TOTAL HRS EARNED	126.0

Class Schedule

CRS	SM1	SM2	NAME - PREREQUISITE	CR	SEC	TIME	INSTRUCTOR	ROOM
POLS		84	SEE HIST 84					
POLS	87	88	SPECIAL PROBLEMS PRE CONSENT	1-3		**	FLUNO	R315
POLS	89	90	THESIS - SEMINAR PRE CONSENT	1-3		7 8 T	FLUNO	R103
POLS		98	HONORS THESIS PRE CANDIDATE FOR MAJOR HONORS	3		**	FLUNO	R315
			PSYCHOLOGY					
PSYC	11		PRINCIPLES	3	A	10 MW 11 F	EACKER	S120
					B	9 MTTH	FOGARTY	M311
PSYC		11	PRINCIPLES	3	A	8 M 9 WF	EACKER	S120
					B	9 MTTH	FOGARTY	M311
PSYC		12	EXP ANAL BEHAVIOR PRE 11	3		9 MTTH	MEYER	B408
					LAB W	1 2 MW	MEYER	
					LAB X	3 4 MW	MEYER	
					LAB Y	1 2 TTH	MEYER	
					LAB Z	3 4 TTH	MEYER	
PSYC	37		SEE ECON 37					
PSYC		43	SOCIAL PSYCHOLOGY PRE 11 OR SOC 3. SAME AS SOC 43	3		11 MTTH	CHERTOK	B408
PSYC	53		ABNORMAL PSYCH PRE 11 - JR	3	A	1 MWF	FOGARTY	B208
					B	10 TTHF	FOGARTY	B308
PSYC		54	EDUCATIONAL PSYCH PRE 11 - JR. FOR TEACHER CANDIDATES	3	A	1 MWF	FOGARTY	B208
					B	10 TTHF	FOGARTY	B308
PSYC	63		CHILD PSYCHOLOGY PRE 11	3		11 MTTH	MEYER	B208
PSYC		67	TESTS - MEASUREMNTS PRE 37	3		1 2 TTH	EACKER	B302
PSYC	71		PERSONALITY THEOR PRE CONSENT	3		3 MWF	MEYER	B302
PSYC	80		THEOR OF LEARNING PRE CONSENT	3		2 MWF	MEYER	B302
PSYC	81		SEMINAR MOTIVATION PRE CONSENT	3		2/30 3 TTH	EACKER	B302
PSYC		82	SEMINAR PRE CONSENT	3		1 MWF	EACKER	B302
PSYC	83		SEM HIST-SYST PSYCH	3		1 MWF	EACKER	B302
PSYC	96	96	THESIS	1		**	FOGARTY	R306
PSYC		98	HONORS THESIS PRE CANDIDATE FOR FOR MAJOR HONORS	3		**	FOGARTY	R306

College Directory

COOPER SUSAN JEAN		FR	ANDERSON HALL SEC C	JA5-9898	157 TROY	JUNEAU	ALAS	99801
COPELAND CHARLES M		JR	1005 ISAACS	JA5-3780	944 BRYANT	CHICO	CALIF	
COPELAND THOMAS A		FR	LYMAN HOUSE SEC E	JA5-9887	5460 BONITA PL	TUCSON	ARIZONA	
CORNUE MARTHA JANE		SR	PRENTISS HALL SEC B	JA5-8972	140 KING	WALLACE	IDAHO	
COTTLE JAMES SIDNEY		SR	715 ESTRELLA	JA5-6542	12300 MELODY LN	LOS ALTOS	CALIF	
COURTNAGE JOHN CREEGAN		FR	JEWETT HALL 403	JA5-9839	2213 FORAKER DR	ANCHORAGE	ALAS	99503
COWEN MARILYN JEAN		SR	PRENTISS HALL SEC A	JA5-3240	1876 OAK KNOLL CT	LAKE OSWEGO	ORE	
CRAIG PATRICK EDGAR		FR	LYMAN HOUSE SEC E	JA5-9887	728 E WHITMAN	WALLA WALLA	WASH	99362
CRAMER SYDNEY ANN		SO	PRENTISS HALL SEC E	JA5-8642	608 OAK	SANDPOINT	IDAHO	83864
CRAVEN DAVID ROGER		JR	949 ISAACS	JA5-5653	3839 DIXON PL	PALO ALTO	CALIF	
CRAVEN HENRY TRUXTON		SR	949 ISAACS	JA5-5653	KEELER RD	BRIDGEWATER	CONN	
CRAWFORD KAREN HALEEN		FR	SAME AS HOME ADDRESS	JA9-2240	2037 CRAWFORD DR	WALLA WALLA	WASH	99362
CROSBY JOHN LLOYD		FR	JEWETT HALL 402	JA5-9839	9915 S TACOMA WAY	TACOMA	WASH	
CROWELL ROBERT IKUA		FR	JEWETT HALL 210	JA5-9979	BOX 598	KAUAI WAIMEA	HAWAII	96796
CRUTCHER DONNA JEANE		SO	SAME AS HOME ADDRESS	JA5-8763	1920 LOUBECK	WALLA WALLA	WASH	
CUDNEY CHARLES HENRY		FR	JEWETT HALL 406	JA5-9839	2009 S 10TH	LAS VEGAS	NEV	89105
CULHAM WILLIAM M	M	JR	809 VALENCIA	JA5-4187	113 SE 10TH	PENDLETON	ORE	
CULLEY JAMES ROBERT		JR	949 ISAACS	JA5-5653	BOX 727	WEED	CALIF	
CUMMINS GALE CATHERINE	M	SP	SAME AS HOME ADDRESS	JA5-8667	610 TAYLOR	WALLA WALLA	WASH	99362
CUNNINGHAM PAUL ALAN		FR	JEWETT HALL 325	JA5-9878	1914 MUKILTEO	EVERETT	WASH	98202
CUNNINGHAM TIMOTHY J		SO	1005 ISAACS	JA5-3780	3353 COTTAGE WAY 95	SACRAMENTO	CALIF	95825
CURFMAN GEORGE HARDIN		SO	949 ISAACS	JA5-5653	20 ELM	DENVER	COLO	80220
CURRENT NANCY		FR	PRENTISS HALL SEC B	JA5-8972	9724 MERCERWOOD DR	MERCER ISL	WASH	98040
CUSACK CHARLES ROOS JR		SO	715 ESTRELLA	JA5-6542	4111 LOS COCHES WY	SACRAMENTO	CALIF	95825
CZINGER JOHN LOUIS III		FR	LYMAN HOUSE SEC B	JA5-9988	3142 KEMPTON DR	LOS ALAMITOS	CALIF	
DAINES HOLLIS LOUISE		SR	PRENTISS HALL SEC G	JA5-8792	3509 WOOD ACRES DR	BOISE	IDAHO	83702
DALE FORREST PAUL		SO	31 WHITMAN	JA5-3426	1104 WASHINGTON	TOPPENISH	WASH	
DALLAIRE MARSHA ANN		FR	ANDERSON HALL SEC B	JA5-9896	1401 PERKINS	RICHLAND	WASH	99352
DALY KATHLEEN DIANA		FR	ANDERSON HALL SEC F	JA5-9946	9281 56TH S	SEATTLE	WASH	98118
DANCE DOUGLAS ELDRED		FR	JEWETT HALL 428	JA5-9788	110 EL MONTE	CONCORD	CALIF	94521
DANIELSON SUE JEAN		SO	PRENTISS HALL SEC G	JA5-8792	1424 E BROADWAY	MOUNT VERNON	WASH	98273
DANISHEK STEPHAN D		SO	716 BOYER	JA5-3822	4714 NE 36TH	SEATTLE	WASH	98105
DAUGHERTY EILEEN P		SO	PRENTISS HALL SEC C	JA5-8232	3026 NE 92ND	SEATTLE	WASH	98115
DAUGHERTY PATRICIA A		JR	PRENTISS HALL SEC F	JA5-8592	846 GREENWICH PL	PALO ALTO	CALIF	
DAUGHTERS EUGENE A JR	M	JR	19 E BIRCH	JA5-1227	1556 30TH WEST	SEATTLE	WASH	98199
DAVIDSON CHRISTOPHER		SR	429 LINCOLN	JA9-5336	637 WARM SPRINGS	BOISE	IDAHO	
DAVIES HEATHER MARIE		FR	ANDERSON HALL SEC F	JA5-9946	1818 HOLBROOK	EVERETT	WASH	98201
DAVIS BARBARA JOAN		SP	ANDERSON HALL SEC C	JA5-9898	16751 17TH NW	SEATTLE	WASH	98177
DAVIS BRUCE VANARSDALE		FR	JEWETT HALL 416	JA5-9839	7662 SE 22ND	MERCER ISL	WASH	98040
DAVIS JOHN M JR		SR	715 ESTRELLA	JA5-6542	7662 SE 22ND	MERCER ISLAND	WASH	
DAVIS PAUL FREDERIC		FR	JEWETT HALL 312	JA5-9962	2722 S LINCOLN	SPOKANE	WASH	99203
DAWSON ELIZABETH ANN		JR	PRENTISS HALL SEC B	JA5-8972	RT 1 BOX 42	JOSEPH	ORE	97846
DAWSON MARY		SR	PRENTISS HALL SEC G	JA5-8972	RT 1 BOX 42	JOSEPH	ORE	97846
DAY MICHAEL JUDD		SO	716 BOYER	JA5-3822	718 N GEE	TACOMA	WASH	
DAYTON KENNETH ALLAN		JR	925 ISAACS	JA5-8443	1318 LAKESIDE S	SEATTLE	WASH	98144
DEAN WILLIAM WENDELL		SO	715 ESTRELLA	JA5-6542	1617 SPRUCE	S PASADENA	CALIF	
DEBUTTS BOBETTE		FR	ANDERSON HALL SEC E	JA5-9846	3243 EVERGREEN PT RD	BELLEVUE	WASH	98004
DECHERT HEDY SUSAN		SR	PRENTISS HALL SEC G	JA5-8792	110 W OTIS RD	BARRINGTON	ILL	60010
DECOLAN RALPH RODNEY		SO	LYMAN HOUSE SEC A	JA5-9847	3014 F	VANCOUVER	WASH	98663
DEGRASSE MICHAEL E		FR	JEWETT HALL 319	JA5-9962	725 YAKIMA	WENATCHEE	WASH	99801
DELZELL DAVID ATHOL		FR	JEWETT HALL 202	JA5-9978	65115 VIEWS RD	TACOMA	WASH	98407
DEMEULES JAMES HEAD		JR	1005 ISAACS	JA5-3780	6815 44TH PL NE	SEATTLE	WASH	98115
DEMMY STUART DOUGLAS		FR	JEWETT HALL 430	JA5-9788	2904 NE 46TH	PORTLAND	ORE	97213
DENNEN RODGERS TAYLOR		FR	JEWETT HALL 109	JA5-9949	1139 23 E	SEATTLE	WASH	98102
DENNIS WILLIAM G		JR	716 BOYER	JA5-3882	5167 S CRESTON	SEATTLE	WASH	98178
DERMOND DONNA CAROLYN		SO	PRENTISS HALL SEC G	JA5-8792	805 N FOOTE	OLYMPIA	WASH	
DESHLER DONALD DAVID		SO	715 ESTRELLA	JA5-6542	615 W MERCURY	BUTTE	MONT	59701
DETWILER ANNEKE JAN		FR	ANDERSON HALL SEC E	JA5-9846	10202 SE 28TH	BELLEVUE	WASH	98004
DIBBLE WAYNE ROBERT		SO	LYMAN HALL SEC B	JA5-9988	BOX 16	GROVELAND	MASS	
DICKEL CHARLES TIMOTHY		SO	925 ISAACS.	JA5-8443	850 NW POWHATAN TERR	PORTLAND	ORE	97210
DION JANE SHAW		SO	PRENTISS HALL SEC C	JA5-8232	818 STUART	HELENA	MONT	
DITMARS FRANK R JR		FR	JEWETT HALL 401	JA5-9839	505 PALOS VERDES DR W	PALOS VERDES	CALIF	90275
DOCHEZ MARC WELCH		FR	JEWETT HALL 217	JA5-9979	3959 LA DONNA	PALO ALTO	CALIF	94306
DORAN GARRETT JOHN		FR	JEWETT HALL 106	JA5-9949	912 6TH	SANTA MONICA	CALIF	94103
DOUGLAS DAVID MICHAEL		SR	401 E WHITMAN	-	412 BIRCH	RICHLAND	WASH	99352
DREHER ANNA LESLIE		JR	PRENTISS HALL SEC G	JA5-8792	2305 DALLAS	RICHLAND	WASH	
DRUMMOND HOWARD B		JR	1005 ISAACS	JA5-3780	141 NORTH E	MISSOULA	MONT	59801
DUNCAN DIANA LYNN		SO	PRENTISS HALL SEC B	JA5-8972	2154 MCMILLAN	EUGENE	OREGON	
DUPREE JOHN ROBERT		FR	1005 ISAACS	JA5-3780	306 WEST	WAITSBURG	WASH	
DUSENBERY THOMAS J		FR	JEWETT HALL 313	JA5-9962	12122 S E 21ST	BELLEVUE	WASH	98004
DWIGGINS MICHAEL BELL		SO	949 ISAACS	JA5-5653	3 STANHOPE PLACE	LONDON W2	ENGLAND	
DWIGGINS STEVEN FRANK		SO	949 ISAACS	JA5-5653	3 STANHOPE PLACE	LONDON W2	ENGLAND	
DYE DAVID GARY		SO	715 ESTRELLA	JA5-6542	1261 C SW	EPHRATA	WASH	
EDDY JOHN WHITTEMORE		FR	JEWETT HALL 409	JA5-9839	4515 W RUFFNER	SEATTLE	WASH	98199
EAGLESON JEANNE ANNE		SO	PRENTISS HALL SEC C	JA5-8232	625 S PALOUSE	WALLA WALLA	WASH	99362
EASTON ANN LOUISE		FR	ANDERSON HALL SEC E	JA5-9846	7704 OXON HILL	OXON HILL	MD	20021
EBY BEN WILDER		SP	SITTNER HALL COLL PL	-	10312 W COURT	PASCO	WASH	
EDWARDS GEORGE MICHAEL		SR	16 S PARK 4	-	1135 ALVARADO	WALLA WALLA	WASH	99362
EICHELBERGER DONALD N		FR	JEWETT HALL 217	JA5-9979	522 32ND S	SEATTLE	WASH	98144
ELDERKIN RICHARD H		JR	949 ISSACS	JA5-5653	2537 STATE	BUTTE	MONT	59701
ELLIS WILLIAM L		SO	1005 ISAACS	JA5-3780	E9404 MAIN	SPOKANE	WASH	
ELLS WILLIAM ARTHUR		SO	715 ESTRELLA	JA5-6542	BOX 164	CUTTEN	CALIF	
ELMORE RICHARD F		SR	135 S PARK	-	1706 N EASTMONT	E WENATCHEE	WASH	
EMEL JOHN ERIC		SR	633 MILITARY RD	JA5-8443	7321 172 SW	EDMONDS	WASH	
EMPEY SUSAN LEE		SO	PRENTISS HALL SEC B	JA5-3582	1009 W 21ST	SPOKANE	WASH	
ENGLAND LAUREL ANN		FR	ANDERSON HALL SEC E	JA5-9846	2504 BARGE	YAKIMA	WASH	98902
ERICKSON KENT LYNN		SO	LYMAN HOUSE SEC C	JA5-9987	616 S 117TH	TACOMA	WASH	98444
ERNST CHARLES JOHN		SO	949 ISAACS	JA5-5653	1055 LEMON	MENLO PARK	CALIF	
ESSER JILL		SR	PRENTISS HALL SEC A	JA5-3240	125 N PERCIVAL	OLYMPIA	WASH	
ESTRIN ROBERT NORMAN		SO	1005 ISAACS	JA5-3780	833 S CITRUS	LOS ANGELES	CALIF	90036
EVANS CAROLYN JANE		SR	PRENTISS HALL SEC A	JA5-3240	501 LAKEVIEW BLVD	SANDPOINT	IDAHO	83864
EVANS MICHAEL JOHN		JR	LYMAN HOUSE SEC A	JA5-9847	1034 A 48TH	LOS ALAMOS	NEW MEX	87544
EVANS SUSAN ETHEL		SO	PRENTISS HALL SEC B	JA5-8972	4774 NE 180TH	SEATTLE	WASH	

WHITMAN COLLEGE BUDGET FORECASTING PROGRAM

Each year Whitman College prepares ten year forecasts for the operating budget. In the past this has been done by hand computation. These calculations of income and expense are quite simple, but take considerable time. Only a few variations of income and expense could be calculated because of the time involved in making calculations.

This problem of calculation time was solved by writing a computer program to do the computation of income and expense. The budget forecasting program reads the input data, performs the calculations and prints the results in approximately six minutes. This short time makes it feasible to recompute income and expense for any change of input data. Thus input data can be varied until a satisfactory solution can be obtained.

In writing a program of this type there are several planning decisions to be made. An example is the consideration of endowment growth. Endowment growth can be projected as a percentage growth or an estimate can be made of the endowment principal for each of the ten years. It was decided to use the latter method for the budget forecasting program. In contrast, the program calculates scholarship costs based on a single piece of input data called the scholarship ratio. The scholarship cost ratio is multiplied by tuition income for each of the ten years to yield scholarship costs. The choice of method for any item of income or expense depends on the particular needs of a college. One can modify any part of this program to meet particular needs.

The following items are input data to the program:

Income Data	Fortran Variable	
Number of students	A	There are eleven entries for each of these items, one for each year of the forecast.
Number of Resident Students	AA	
Number who board only	AAA	
Research expenditures	B	
Tuition and Fees	C	
Room Charges	D	
Board Charges	E	
Endowment Principal	F	
Gift Income	P	
Research overhead ratio	G	
Endowment yield	O	

Cost Data	Fortran Variable	
Faculty Salary Increment Ratio	D	11 entries
Special costs	E	
Number of Faculty	P	
First year average faculty salary	F	
Cost of living factor(non-salary items)	H	
Administrative Salary Increment	HH	
Ratio of scholarship costs to tuition fees	SF	
Contingency	CØ	
Faculty Fringe Benefits Ratio	FFB	11 entries
Administrative Fringe Benefits Ratio	AFB	

All costs other than faculty salaries, faculty fringe benefits, scholarship costs, special costs, and contingency are treated as budget categories. Each budget category is split into a salary and a nonsalary part. Further, a budget category is considered as independent of enrollment growth or directly proportional to enrollment growth. There is a provision for adding a jump in either the salary or non-salary part of any budget for any year. The input data for the budget categories are the costs previously mentioned and the salary and non-salary parts for each budget category for the first year.

The nonsalary variables are X(J,I) and the salary variables are Y(J,I) where J is the budget category and varies from 1 to 12 and I is the year and varies from 1 to 11. The number of budget categories which are independent of enrollment growth is arbitrary and can be changed. Presently, the first five budget categories are independent of enrollment and the last seven are directly proportional to enrollment.

The program calculates for each of 11 years:

research income
tuition and fees income
room income
board income
endowment income
total income
student faculty ratio
average faculty salaries
total faculty salaries
faculty fringe benefits
scholarship funds
budgets for all budget categories
total expenditures
surplus
surplus/student
expense/student

All input data and calculated results are printed. There is one piece of data per card with a format of F 10.0. The order of input can be determined by studying the Fortran listings.

```
C       WHITMAN COLLEGE BUDGET FORECAST PROGRAM
C       VERSION 2
C
C       FIRST LINK OF THREE LINK PROGRAM
C
      DIMENSION A(11),B(11),C(11),D(11),E(11),F(11),Q(11),R(11)
      DIMENSION S(11),T(11),U(11),V(11),W(11),X(12,11),Y(12,11),Z(11)
      DIMENSION P(11)
      DIMENSION AA(11), AAA(11)
      DIMENSION FS(11)
      DIMENSION FFB(11), AFB(11)
      COMMON A,B,C,D,E,F,Q,R,S,T,U,V,W,X,Y,Z,P
      COMMON L,K
      COMMON H, HH, SF, CO
      COMMON FFB, AFB
      COMMON FS
    1 FORMAT (F10.0)
    3 FORMAT(1H1,23X,8HRESIDENT,  1X, 12HSTUDENTS WHO)
    4 FORMAT (     13X, 8HSTUDENTS, 3X, 8HSTUDENTS, 3X, 10HBOARD ONLY,
     1 3X, 8HRESEARCH, 2X,
     2 12HTUITION+FEES,  6X, 4HROOM,  7X, 5HBOARD,  2X, 11HGIFT INCOME,
     3 2X, 9HENDOWMENT)
    5 FORMAT( 5HOYEAR, I3, 6( 7X, F10.0))
    6 FORMAT( 24HORESEARCH OVERHEAD RATIO, F10.2/
     1              16H ENDOWMENT YIELD,10X, F10.4)
    7 FORMAT( 1HO,11X, 15HRESEARCH INCOME, 3X, 19HTUITION+FEES INCOME,
     1 2X, 11HROOM INCOME,
     2 5X, 12HBOARD INCOME, 3X, 16HENDOWMENT INCOME, 2X,12HTOTAL INCOME)
    8 FORMAT( 1H1, 8X, 23HFACULTY SALARY INCREASE, 5X, 14HNO. OF FACULTY
     1,5X, 21HSTUDENT FACULTY RATIO)
    9 FORMAT( 5HOYEAR, I3, 7X, F10.3, 12X, F10.1, 12X, F10.2)
   10 FORMAT(                                    / 22H COST OF LIVING FACTO
     1R, 5X, F10.3/ 27H ADMINISTRATIVE SALARY INC.,     F10.3/
     218H SCHOLARSHIP RATIO, 9X, F10.3)
   11 FORMAT ( 5HOYEAR, I3, 9( 2X, F10.0))
   12 FORMAT (5HOYEAR, I3,5X, F10.3,13X, F10.3)
   13 FORMAT (1HO, 16X, 7HFACULTY, 15X, 14HADMINISTRATION/
     1 17X, 15HFRINGE BENEFITS,  7X, 15HFRINGE BENEFITS /
     2 17X, 5HRATIO, 17X, 5HRATIO)
      K=12
      L=11
      DO 101 J = 1,K
      DO 101 I = 1,L
      X(J,I) = 0
  101 Y(J,I) = 0
      PRINT 3
      PRINT 4
      DO 100 I=1,L
      READ 1,A(I),AA(I),AAA(I), B(I),C(I),D(I),E(I),F(I),P(I)
  100 PRINT 11,I, A(I),AA(I),AAA(I),B(I), C(I), D(I), E(I), P(I), F(I)
      READ 1,G,  0
      PRINT 6,G,  0
      PRINT 7
      DO 21 I=1,L
      Q(I)=B(I)*G
```

```
      R(I)=C(I)*A(I)
      S(I) = D(I)*AA(I)
      T(I) = E(I)*AAA(I)
      U(I)=F(I)*O
      V(I)=Q(I)+R(I)+S(I)+T(I)+U(I)+P(I)
   21 PRINT 5,I, Q(I),R(I),S(I),T(I),U(I),V(I)
      DO 32 I=1,L
      READ 1,D(I)
   32 READ 1,E(I)
      READ 1,F(1)
      READ 1,(P(I),I=1,L)
      READ 1,   H,HH,SF
      READ 1,CO
      READ 1,(FFB(I), I=1,L)
      READ 1,(AFB(I), I=1,L)
      DO 30 I= 1, L
   30 FS(I) = A(I)/P(I)
      PRINT 8
      DO 33 I = 1,L
   33 PRINT 9, I, D(I), P(I), FS(I)
      PRINT 10,     H, HH, SF
      PRINT 13
      DO 600 I=1,L
  600 PRINT 12, I, FFB(I), AFB(I)
      CALL LINK (FOR2)
      END
```

```
C       WHITMAN COLLEGE BUDGET FORECAST PROGRAM
C       VERSION 2
C
C       SECOND LINK OF THREE LINK PROGRAM
C
      DIMENSION A(11),B(11),C(11),D(11),E(11),F(11),Q(11),R(11)
      DIMENSION S(11),T(11),U(11),V(11),W(11),X(12,11),Y(12,11),Z(11)
      DIMENSION P(11)
      DIMENSION FFB(11), AFB(11)
      DIMENSION FS(11)
      COMMON A,B,C,D,E,F,Q,R,S,T,U,V,W,X,Y,Z,P
      COMMON L,K
      COMMON H, HH, SF, CO
      COMMON FFB, AFB
      COMMON FS
    1 FORMAT (F10.0)
    2 FORMAT ( I2, I2, F10.0, F10.0 )
  104 FORMAT( 1H1, 9X, 24HAVERAGE FACULTY SALARIES)
  105 FORMAT( 5HOYEAR, I3, 10X, F10.0)
      DO 34 J=1,K
   34 READ 1,X(J,1)
      DO 35 J=1,K
   35 READ 1,Y(J,1)
  201 READ 2,M,N,AZ1,AZ2
      IF (M-K) 300,300,200
  300 IF (N-L) 301,301,200
  301 X(M,N)=AZ1
      Y(M,N)=AZ2
      GO TO 201
  200 DO 31 I= 1, L
   31 P(I) = FS(I)
      DO 41 I=2,L
   41 F(I)=F(I-1)*D(I)
      PRINT 104
      DO 51 I=1,L
   51 PRINT 105, I, F(I)
      DO 42 I=1,L
      F(I)=A(I)/ P(I)*F(I)
   42 Q(I)=F(I)*FFB(I)
      DO 49 J=1,K
      BBB=A(1)
      DO 44 I=2,L
      X(J,I)=X(J,I-1)*H + X(J,I)
      Y(J,I)=Y(J,I-1)*HH + Y(J,I)
   44 Y(J,I) = Y(J,I) +Y(J,I) * AFB(I)
      X(J,1) = X(J,1) + Y(J,1)
      DO 43 I = 2,L
      X(J,I)=X(J,I)+Y(J,I)
      IF (X(J,I-1)) 50,50,45
   50 IF (X(J,I)) 43,43,60
   60 Y(J,I)=X(J,I)
      BBB=A(I)
      GO TO 43
   45 Y(J,I)=X(J,I)*A(I)/BBB
   43 CONTINUE
```

```
   49 CONTINUE
C       AT THIS POINT THE X ARRAY CONTAINS NONSALARY+SALARY WITH NO GROWTH FACTU
C       AT THIS POINT THE Y ARRAY CONTAINS NONSALARY+SALARY TIMES GROWTH FACTOR
C       HERE GROWTH FACTOR MEANS ENROLLMENT
      DO 46 I=1,L
   46 R(I)=R(I)*SF
      DO 47 I=1,L
      S(I)=X(1,I)+X(2,I)+X(3,I)+X(4,I)+X(5,I)
   47 T(I)=Y(6,I)+Y(7,I)+Y(8,I)+Y(9,I)+Y(10,I)+Y(11,I)+Y(12,I)
      DO 48 I=1,L
      U(I)=B(I)+F(I)+Q(I)+R(I)+S(I)+T(I)+E(I)+CO
   48 W(I)=V(I)-U(I)
      CALL LINK(FOR3)
      END
```

```
C       WHITMAN COLLEGE BUDGET FORECAST PROGRAM
C       VERSION 2
C
C       THIRD LINK OF THREE LINK PROGRAM
C
      DIMENSION A(11),B(11),C(11),D(11),E(11),F(11),Q(11),R(11)
      DIMENSION S(11),T(11),U(11),V(11),W(11),X(12,11),Y(12,11),Z(11)
      DIMENSION P(11)
      DIMENSION FFB(11), AFB(11)
      DIMENSION  ZZ(11)
      COMMON A,B,C,D,E,F,Q,R,S,T,U,V,W,X,Y,Z,P
      COMMON L,K
      COMMON H, HH, SF, CO
  106 FORMAT(1HO,10X,13HRESEARCH EXP., 3X,22HTOTAL FACULTY SALARIES, 3X,
     1 18HFACULTY RETIREMENT, 3X, 17HSCHOLARSHIP FUNDS, 5X,
     2 13HSPECIAL COSTS)
  107 FORMAT( 5HOYEAR, I3, 4X, F10.0, 7X, F10.0,12X, F10.0,12X, F10.0,
     112X, F10.0)
  108 FORMAT( 12HOCONTINGENCY, 5X, F10.0)
  109 FORMAT( 1H1, 18X, 51HBUDGET CATEGORIES NOT SUBJECT TO ENROLLMENT
     1GROWTH/   15X, 4HNO.1, 10X, 4HNO.2, 10X, 4HNO.3, 10X, 4HNO.4,
     2 10X, 4HNO.5)
  110 FORMAT( 5HOYEAR, I3, 3X, F10.0, 4X, F10.0, 4X, F10.0, 4X, F10.0,
     1 4X, F10.0)
  111 FORMAT( 1HO, 25X, 46HBUDGET CATEGORIES SUBJECT TO ENROLLMENT GROWT
     1H/ 15X, 4HNO.6, 10X, 4HNO.7, 10X, 4HNO.8, 10X, 4HNO.9, 9X, 5HNO.10
     1, 9X, 5HNO.11, 9X, 5HNO.12)
  112 FORMAT( 5HOYEAR, I3, 3X, F10.0, 4X, F10.0, 4X, F10.0, 4X, F10.0,
     1 4X, F10.0, 4X, F10.0, 4X, F10.0)
  113 FORMAT (1H1, 11X, 12HTOTAL INCOME, 5X, 18HTOTAL EXPENDITURES,  5X,
     1 7HSURPLUS,  5X, 15HSURPLUS/STUDENT,  5X, 15HEXPENSE/STUDENT)
  114 FORMAT (5HOYEAR, I3, 5(7X, F10.0))
      PRINT 106
      DO 52 I=1,L
   52 PRINT 107, I, B(I), F(I), Q(I), R(I), E(I)
      PRINT 108, CO
      PRINT 109
      DO 53 I=1,L
   53 PRINT 110, I, X(1,I),X(2,I),X(3,I),X(4,I),X(5,I)
      PRINT 111
      DO 54 I=1,L
   54 PRINT 112, I,Y(6,I),Y(7,I), Y(8,I),Y(9,I),Y(10,I),Y(11,I),Y(12,I)
      PRINT 113
      DO 60 I=1,L
      Z(I)=W(I)/A(I)
      ZZ(I) = U(I) / A(I)
   60 PRINT 114, I,V(I), U(I), W(I), Z(I), ZZ(I)
      CALL EXIT
      END
```

fictitious data

	STUDENTS	RESIDENT STUDENTS	STUDENTS WHO BOARD ONLY	RESEARCH	TUITION+FEES	ROOM	BOARD	GIFT INCOME	ENDOWMENT
YEAR 1	970.	610.	610.	15000.	1180.	320.	490.	85500.	4860000.
YEAR 2	980.	620.	630.	15000.	1580.	360.	520.	85500.	5103000.
YEAR 3	980.	620.	630.	15000.	1580.	360.	520.	85500.	5358000.
YEAR 4	990.	630.	650.	15000.	1580.	360.	520.	85500.	5626000.
YEAR 5	1000.	640.	670.	15000.	1580.	380.	550.	85500.	5907000.
YEAR 6	1000.	640.	670.	15000.	1780.	380.	550.	90000.	6203000.
YEAR 7	1050.	660.	690.	15000.	1780.	380.	550.	90000.	6513000.
YEAR 8	1200.	750.	790.	15000.	1780.	400.	580.	90000.	6838000.
YEAR 9	1200.	750.	790.	15000.	1780.	400.	580.	90000.	7180000.
YEAR 10	1210.	760.	800.	15000.	1780.	400.	580.	90000.	7539000.
YEAR 11	1220.	770.	810.	15000.	1780.	400.	580.	90000.	7915000.

111

RESEARCH OVERHEAD RATIO 1.32
ENDOWMENT YIELD .0600

	RESEARCH INCOME	TUITION+FEES INCOME	ROOM INCOME	BOARD INCOME	ENDOWMENT INCOME	TOTAL INCOME
YEAR 1	19800.	1144600.	195200.	298900.	291600.	2035600.
YEAR 2	19800.	1548400.	223200.	327600.	306180.	2510680.
YEAR 3	19800.	1548400.	223200.	327600.	321480.	2525980.
YEAR 4	19800.	1564200.	226800.	338000.	337560.	2571860.
YEAR 5	19800.	1580000.	243200.	368500.	354420.	2651420.
YEAR 6	19800.	1780000.	243200.	368500.	372180.	2873680.
YEAR 7	19800.	1869000.	250800.	379500.	390780.	2999880.
YEAR 8	19800.	2136000.	300000.	458200.	410280.	3414280.
YEAR 9	19800.	2136000.	300000.	458200.	430800.	3434800.
YEAR 10	19800.	2153800.	304000.	464000.	452340.	3483940.
YEAR 11	19800.	2171600.	308000.	469800.	474900.	3534100.

	FACULTY SALARY INCREASE	NO. OF FACULTY	STUDENT FACULTY RATIO
YEAR 1	1.050	67.0	14.47
YEAR 2	1.050	68.0	14.41
YEAR 3	1.050	70.0	14.00
YEAR 4	1.050	75.0	13.20
YEAR 5	1.050	76.0	13.15
YEAR 6	1.100	76.0	13.15
YEAR 7	1.050	78.0	13.46
YEAR 8	1.050	84.0	14.28
YEAR 9	1.050	86.0	13.95
YEAR 10	1.050	90.0	13.44
YEAR 11	1.050	98.0	12.44

COST OF LIVING FACTOR	1.015
ADMINISTRATIVE SALARY INC.	1.044
SCHOLARSHIP RATIO	.250

	FACULTY FRINGE BENEFITS RATIO	ADMINISTRATION FRINGE BENEFITS RATIO
YEAR 1	.070	0.000
YEAR 2	.070	0.000
YEAR 3	.070	0.000
YEAR 4	.070	0.000
YEAR 5	.070	0.000
YEAR 6	.070	0.000
YEAR 7	.070	0.000
YEAR 8	.070	0.000
YEAR 9	.070	0.000
YEAR 10	.070	0.000
YEAR 11	.070	0.000

AVERAGE FACULTY SALARIES

YEAR 1	10073.
YEAR 2	10576.
YEAR 3	11105.
YEAR 4	11660.
YEAR 5	12243.
YEAR 6	13468.
YEAR 7	14141.
YEAR 8	14848.
YEAR 9	15591.
YEAR 10	16370.
YEAR 11	17189.

113

	RESEARCH EXP.	TOTAL FACULTY SALARIES	FACULTY RETIREMENT	SCHOLARSHIP FUNDS	SPECIAL COSTS
YEAR 1	15000.	674891.	47242.	286150.	0.
YEAR 2	15000.	719212.	50344.	387100.	100000.
YEAR 3	15000.	777383.	54416.	387100.	0.
YEAR 4	15000.	874556.	61218.	391050.	100000.
YEAR 5	15000.	930528.	65136.	395000.	0.
YEAR 6	15000.	1023581.	71650.	445000.	0.
YEAR 7	15000.	1103043.	77213.	467250.	100000.
YEAR 8	15000.	1247287.	87310.	534000.	100000.
YEAR 9	15000.	1340833.	93858.	534000.	100000.
YEAR 10	15000.	1473358.	103135.	538450.	100000.
YEAR 11	15000.	1684539.	117917.	542900.	100000.

CONTINGENCY 0.

BUDGET CATEGORIES NOT SUBJECT TO ENROLLMENT GROWTH

	NO.1	NO.2	NO.3	NO.4	NO.5
YEAR 1	589710.	1066470.	100000.	0.	0.
YEAR 2	615367.	1182467.	101500.	0.	0.
YEAR 3	642149.	1200204.	103022.	110000.	0.
YEAR 4	670104.	1218207.	104567.	111940.	0.
YEAR 5	699286.	1236480.	106136.	113921.	0.
YEAR 6	729746.	1255027.	107728.	115946.	0.
YEAR 7	761543.	1273852.	109344.	118015.	0.
YEAR 8	794734.	1292960.	110984.	120130.	0.
YEAR 9	829380.	1312354.	112649.	122292.	0.
YEAR 10	865546.	1332040.	114338.	124502.	0.
YEAR 11	903299.	1352020.	116054.	126761.	0.

BUDGET CATEGORIES SUBJECT TO ENROLLMENT GROWTH

	NO.6	NO.7	NO.8	NO.9	NO.10	NO.11	NO.12
YEAR 1	0.	0.	0.	0.	0.	0.	0.
YEAR 2	0.	0.	0.	0.	0.	0.	0.
YEAR 3	0.	0.	0.	0.	0.	0.	0.
YEAR 4	0.	0.	0.	0.	0.	0.	0.
YEAR 5	0.	0.	0.	0.	0.	0.	0.
YEAR 6	0.	0.	0.	0.	0.	0.	0.
YEAR 7	150000.	0.	0.	0.	0.	0.	0.
YEAR 8	175657.	0.	0.	0.	0.	0.	0.
YEAR 9	180022.	0.	0.	0.	0.	0.	0.
YEAR 10	186066.	0.	0.	0.	0.	0.	0.
YEAR 11	192335.	0.	0.	0.	0.	0.	0.

	TOTAL INCOME	TOTAL EXPENDITURES	SURPLUS	SURPLUS/STUDENT	EXPENSE/STUDENT
YEAR 1	2035600.	2779463.	-743863.	-766.	2865.
YEAR 2	2510680.	3170991.	-660311.	-673.	3235.
YEAR 3	2525980.	3289276.	-763296.	-778.	3356.
YEAR 4	2571860.	3546645.	-974785.	-984.	3582.
YEAR 5	2651420.	3561489.	-910069.	-910.	3561.
YEAR 6	2873680.	3763680.	-890000.	-890.	3763.
YEAR 7	2999880.	4175262.	-1175382.	-1119.	3976.
YEAR 8	3414280.	4478064.	-1063784.	-886.	3731.
YEAR 9	3434800.	4640391.	-1205591.	-1004.	3866.
YEAR 10	3483940.	4852437.	-1368497.	-1130.	4010.
YEAR 11	3534100.	5150827.	-1616727.	-1325.	4221.

AUSTIN COLLEGE

SYMBOLIC PROGRAMMING SYSTEM

Presented by

David R. Musser
Hoblitzelle Computer Center
Austin College
Sherman, Texas

WESTERN REGION SUMMER MEETING OF COMMON

Denver Hilton Hotel, Denver, Colorado

July 7, 1966

Program Abstract

Title: Austin College Symbolic Programming System

Author: David R. Musser
Austin College
Sherman, Texas 75091

Date:

Users Group Code: 5252

Direct Inquiries to: Mr. Robert F. Randall, Director
Hoblitzelle Computer Center
Austin College
Sherman, Texas 75091
Phone: 214-892-9101 Ext. 40

Subject Classification: 1.1

Description/Purpose:

AC-SPS is an SPS system designed to provide users of a 1620 without a disk-file some of the versatility, speed and convenience of disk operation. This is achieved by modifying an existing SPS system, AFIT SPS (1.1.027 , written by R. L. Pratt), which is itself an improvement of the standard IBM system, especially in terms of speed of operation. Essentially all of the features and capabilities of AFIT SPS have been retained, but additional features are provided: (1) a supervisor program which controls processing by interpreting control cards; (2) load-and-go operation for small programs, which reduces the time required for debugging programs; (3) macro instruction subroutines for input-output operations (READ, ACPT, PNCH, TYPE), which provide an I/O system similar to that of Basic Fortran, greatly simplifying I/O programming.

Specifications:

Storage: 40K or larger (self-adjusting)

Equipment: Card system, TNS, TNF, MF, Auto-divide, Indirect addressing. Requirements for TNS, TNF, MF and Auto-divide could be removed fairly easily by reprogramming.

Language: AFIT SPS. Writeup is in English.

PROGRAM DESCRIPTION

Introduction

Austin College SPS is a modification of AFIT SPS (1.0.027--"AFIT Symbolic Programming System, Modified for 80-80 Listing"). Originally AC-SPS was developed as a modification to IBM SPS, but the resulting system left something to be desired with regard to speed of operation. It was then found that the AFIT SPS system, written by Mr. R. L. Pratt, was much superior to IBM SPS, especially in terms of speed. In fact, AFIT SPS assembles three to four times as fast as IBM SPS, for the following reasons:

A binary search of the opcode and label tables is used.
Indirect addressing and MF, TNF, TNS are used.
A listing deck suitable for listing with a standard 80-80 list board is produced (one listing card per source program card, usually); thus the deck requires only half as long to punch, half as long to list.
During the second pass, the source program is retrieved from memory rather than reread (if it was possible to store the entire source program during pass one.)

Aside from its greater speed, AFIT SPS also provides additional language features, which in a number of instances make programming more convenient.

AC-SPS retains the speed of operation and essentially all the capabilities of AFIT SPS, but makes it more versatile and more convenient to operate by the addition of the following features:

(1) A supervisor program, which controls the processor by interpreting control cards included in the source program deck by the programmer.
(2) Load-and-go operation for short programs. Only a few minor restrictions are placed on the source language permitted when the processor is operated in the load-and-go mode. All essential floating point macro instructions are allowed, assuming that the special load-and-go subroutine deck has been loaded.
(3) Input -output macro instruction subroutines, which provide an I/O system similar to the system of Basic Fortran, thus greatly simplifying I/O programming. These macros are available for either load-and-go operation or inclusion in punched object decks.

The first two of these features combine to give the system some of the power and convenience of a disk SPS processor under the control of a monitor system. The third feature makes SPS a more useful language, as it becomes somewhat more competitive with Fortran with regard to ease of programming (especially for short programs which formerly often required more programming effort for I/O than for actual computation), while retaining its greater versatility.

The present description of AC-SPS will deal only with the more significant aspects of the features which have been added to AFIT SPS. Familiarity with the AFIT SPS system would be helpful, but is not essential to the understanding and operation of AC-SPS, if the reader is familar with the Reference Manual, IBM 1620/1710 Symbolic Programming System, C 26-5600. When the program is submitted to the Users Group Library, a more thorough description of all features will be included in the writeup.

Supervisor program

The use of a supervisor program was motivated mainly by the desire for more convenient operation (especially since more options were provided, making too many to reasonably handle with switch settings). It is also felt that this system can be used to introduce programming students to the supervisor program concept, which, of course, they will need to become familiar with if they later work with a larger machine or a 1620 with a disk-file.

Most, but not all, of the control of the operation of the processor is determined by control cards prepared by the programmer and included in his source deck. In general, all options over which it is felt the programmer should have advance control have been defined as control card options, and a few remaining options have been left under manual control of the console switches. The latter category consists of options which are not often used, and/or which require action by the console operator; e.g. input or error correction from the typewriter, and listing on the typewriter. These options are controlled by Switches 1, 2, and 3 (see the operating instructions). If any of these options is desired, then the processor should be operated with switch 4 on, which causes the processor to halt after each stage of processing so that Switches 1-3 can be reset if necessary. Generally, however, none of these options is desired, and the processor may be operated with all four switches off. In this case, there is usually no need for operator intervention, and the machine does not halt unless there is. This generally means faster, more efficient operation when a number of short programs are being processed.

The supervisor-processor system processes jobs. A job is defined here to be deck of cards composed of either:

	Job card*
Source	Source program deck*, including source program
Program	control cards
Job	Data cards (if the program is to be executed and contains read statements)
	End-of-file card

or

	Job card*
Object	Object program control cards
Program	Object program*
Job	Data cards
	End-of-file card

*Must be present in all jobs of this type.

The control cards are described in the operating instructions. It may be noted here that the simplest possible source program job consists of

Job card
Source program deck, without any source program control cards

For such a job, the program will be assembled into memory, and if no errors are detected, it will be executed. This mode of operation, called the load-and-go mode, is described in the next section. An object deck, a listing deck, etc., may be obtained, and the operation of the system may be otherwise modified, by the inclusion of one or more of the source program control cards.

Similarly, the simplest possible object program job is

Job card
Object program

In this case the object program is loaded under control of the supervisor program loader, which protects the supervisor-processor system, and the subroutines (if they are in memory). Control cards are provided to remove some or all of this protection when necessary, or to allow the loading of a non-AC-SPS assembled object deck.

Load-and-go mode

In this mode of operation no object deck is punched; rather, the object program is loaded into a work area of memory as it is assembled. This saves time and card handling, and permits complete processing of a job, including execution of the object program with data after it has been assembled. This is especially useful when debugging a program.

A limited attempt has been made to protect the supervisor-processor from being damaged by errors in the program during execution, and to make operation as continuous as possible.

(1) The area of memory into which the object program is loaded is initialized to flags and record marks, so that undefined variable errors will not clear memory.

(2) No attempt is made to check for any of the countless possible errors which can cause check-stops, but an easy restart procedure is available when they occur.

(3) The halt statement is modified so that control is returned to the supervisor when it is executed, rather than an actual halt occuring. Also, the input macros provided with the system can detect an end-of-file card and return control to the supervisor. Thus programs terminated with either a halt statement or running out of data cards will not halt the automatic operation of the system.

These and certain other aspects of the load-and-go mode are discussed more fully in the operating instructions.

The remainder of this discussion is concerned with restrictions on the size of a program run in this mode, and minor restrictions on the language permitted in the source program.

Size of the program

On a 40K machine, a program which requires the subroutines is restricted to 7000 digits length; one which does not may be 16000 digits long.

Language

TRA and TCD statements and the DNB statement are not allowed in the load-and-go mode.

Generally, the DORG statement should not be used to define the origin of

the program itself, as the processor automatically assigns it properly, and changing this assignment may result in loading of the object program being inhibited.

It is even more important that absolute addresses in the source program be avoided, since if such appear in instructions then the supervisor-processor may be clobbered when these instructions are executed. (The processor does not check for this.)

The DEND statement must have as its operand the label of the instruction where execution is to begin; if this operand is omitted then execution of the program will be bypassed.

The load-and-go subroutine deck contains most, but not all of the standard floating point subroutines. DIV, ATAN, FSLS and FSRS have not been included. Subroutines which were added to the AFIT system, INC, OUTC, FC and CKPP, have also been deleted.

Input-output macros

The INC and OUTC macros were added to the AFIT SPS system to make I/O programming more convenient than it is using the hardware I/O commands. In AC-SPS these are replaced by an even more powerful set of macros, READ, ACPT, PNCH and TYPE, which provide an I/O system which is similar to the system used in Basic Fortran. No attempt will be made here to describe in detail the usage of these macros. The discussion will be instead limited to a simple example.

Consider the following Fortran statements:

```
      PUNCH 1, A, B, C, I, J, P, Q

    1 FORMAT (F10.4,F10.2,E14.7,I5,I10)
```

Corresponding to these Fortran statements are the following AC-SPS statements:

```
      PNCH F1, A, B,C/I, J, P, Q,
      .
      .
      .
F1    DSA 61004,61002,51407,90600,91000,
```

Assuming that the quantities A,B, etc., were correctly stored, the output produced by the SPS statements would be exactly the same as that produced by the Fortran statements. This example illustrates several aspects of the AC-SPS I/O macros:

1. The operands field contains first the label of a "Format declaration," then any number of labels (or absolute addresses or any other valid form of operand for a macro instruction), separated by commas. Note that the last operand also has a trailing comma, which cannot be omitted. The I/O subroutine itself uses the five digit zero field produced by this comma to determine the end of the list of variables.

2. The "Format declaration" is a labeled DSA statement with five digit codes (these are not addresses) as its operands, representing format specifications. Permitted numeric conversion specifications include:

Code	Corresponding Fortran Specification
5	E
6	F
9	I

There is also a "D" specification, code 4, which stores numbers in fixed point fields on input but inserts a decimal point in numbers output from a fixed point field. Code 7 can be used to obtain the same effect as the non-numeric X specification in Fortran.
This code digit is the first digit of the five digit code. The next two digits specify the width of the field in which the number is contained in external form. The last two digits specify the position of the decimal point for the "D", "E", and "F" specifications. Thus the form of the entire specification is

cwwdd

3. Note the presence of the trailing comma in the format declaration also. This has somewhat the same effect as the right parenthesis in the Fortran Format statement when the number of specifications is less than the number of variables in the list; i.e. a new record (card or typewriter line) is skipped to, and the beginning of the format declaration is returned to for specifications to use with the remaining variables.

4. Format DSA's, like most other constants in an SPS program, cannot be interspersed with instructions in the way that Format statements can in a Fortran program.

The input routine reads free form data; that is, field widths are not observed and numbers to be input may be positioned in any manner, except that they must be separated by one or more blanks. (In fact, the input routine is closely patterned after the input routine in the PDQ Fortran Free Form Subroutines.) The E and F specifications may be used interchangeably to convert the standard E and F forms to floating point fields, and the I and D specifications interchangeably to convert numbers with or without a decimal point (if present, it is ignored) to fixed point fields. The I and D specifications use the dd code digits to determine the length of the field in which to store the numbers.

Use of the output routine is very much the same as use of Basic Fortran output routines, except that when the width of an F specification is exceeded, the number is automatically output in E form. Care must be taken, however, when using the I and D forms that ww is sufficiently large for the length of the field being ~~input.~~ output.

Student Data Processing System at Christian Brothers College Employing an IBM 1620

Brother Jerome David Wegener, Registrar
Christian Brothers College, Memphis, Tennessee

Christian Brothers College is a small all men's college. Last year we began with 950 students, and this year we expect to have about 1,050. We have an unusual situation in that nearly 60% of our students are majoring in engineering or science. It was for the instruction of these students that a 1620 was installed in January of 1962. The computer center is run on a completely open shop basis both for students and faculty. As a member of the faculty and Registrar, I learned Fortran and SPS, and we began using the computer center for records in January of 1963.

The computer center consists of an IBM 1620, Model II, a 407, an 082 sorter, and an 026 key punch with a special Interspersed Gang Punch feature. Previous to this time no machinery at all was used in the Registrar's office. With the exception of a typewriter, all the work was done by hand. We did not graduate from a Unit Record operation, though our present system probably resembles that greatly since it is a completely card-oriented system. The computer is used as a substitute for many of the ordinary hand and machine operations.

During the second semester of 1962-63, we paralleled the hand system with the computerized system, and in the fall of 1963 we went over to the computerized system completely. Besides taking care of the grades and other records, the system is also used to make out bills for each student.

I. INITIATING A STUDENT:

The first step in getting a new student into the system is to prepare an admission card. This is done by information taken from the student's application blank and transcript. When several such cards are available, a special program is used to assign student numbers according to alphabetical order. It places the new student halfway between the numbers of two students who are already in the file. The output of this program is a special card which carries the same information as the admission card, with the addition of the student number. These are then interpreted and kept in a special file containing all students.

By duplicating certain parts of this card, a #1 card is prepared. This is the student's name card, and later carries much additional coded information. For students who have been in school the previous semester, these cards can simply be updated. A special program has been written which updates the year and duplicates all the cards except those of graduating seniors (whose cards are not duplicated). Certain other changes must be made by hand, such as changing the probationary or boarding status of a student, or his major.

With the #1 card as a starting point, the other basic cards are produced by key punching the appropriate information. These include the #2 card, used for billing address; the #8 card, used for parent's or guardian's address; and the #9 card, used for local address and telephone number. A special program produces from the #1 card, the #7 card, which contains the name of the student's major and the high school he attended. Besides being used for addressing labels, all these cards are used in printing the student information form.

II. INITIATING REGISTRATION:

At the present time, the Master Schedule is made out completely by hand. No attempt has been made to computerize this or to do sectioning, mainly because no one has had the time to devote to it.

Once the Master Schedule is made out, the list of teachers is alphabetized, and assigned numbers in order beginning with number one. The program at present allows for 99 instructors. Next the courses are numbered, alphabetically by department, beginning with number one. At present, the program allows for 240 course sections. The above mentioned numbers change from semester to semester and are used mainly by the computer in keeping track of totals. Department numbers and major field numbers remain the same from semester to semester.

With this information, the Master Course cards are made out, one for each section listed on the schedule. These are identical to the Student Grade Card (#3 card) with the exception of putting the teacher's name in place of the student's name. These are punched on right corner cut cards in order to make use of the interspersed gang punch.

Blank Grade Cards are counted out by the computer for each section according to the estimated number of cards for each section. Each group is placed after the proper Master Course Card and these are then gang punched, leaving the place for the student's name and number blank.

III. REGISTRATION:

At the time of registration, the students fill out a single registration card listing their schedule and certain information. With this in hand they proceed to the gym or other designated area, pick up their name card and move around the gym picking up a Student Grade Card for each course that they have listed on their schedule. At the end of the line, the cards are checked to see if that student has made no mistakes. This entire process takes about ten minutes.

The name card is a right-corner cut card, and is kept in front of the student's Grade Cards. These are then gang punched making use of the Interspersed Gang Punch features on the key punch. Once the student's name and number have been punched on all the cards, the name cards are separated from the Grade Cards, and these latter are sorted into alphabetical order by course along with the Master Course Cards, and the temporary class lists are printed. Once copy is given to each instructor for each of his classes, and a complete set to the Dean's office.

The cards are then re-sorted into alphabetical order by student. In this sort, all the cards numbered 1,2,3,7,8, and 9 are used, and the student's schedule and information sheet is printed. One copy is given to the Dean's office and one to the Business Office. A copy of the information sheet is given to each student so that he can report any errors in addresses, or schedule to the Dean's office. The students have ten days in which to finalize their schedules. During this time, any changes are recorded on Change Slips and the appropriate change is made among the student's Grade Cards. At the end of this time, the Final Class Lists are printed, as well as new student information sheets.

After the registration has been finalized, distributions are calculated according to the various categories, such as by Major, by state or country, by year in school, by boarding status, and by high school attended. A special program is used to do this. Listings are also made according to these various categories.

Another program is used to count the number of Grade Cards that each student has, and at the same time determine those who are special students. A new name card is punched which contains this additional information. This is done so that, with all the handling of the cards, we can check to see that none is lost.

IV. AT THE MARKING PERIODS:

A week or so before marks are due, the Grade Cards are sorted into order by courses and distributed to the teachers along with the Grade Sheets. The teachers record the marks on the cards and on the sheets, and return them to the Registrar's office. The cards are hand sorted into groups by grade, and the computer is used to gang-punch the grades into the cards. The F's are done first so that the failure list can be printed.

The Grade Cards are then sorted into order by student, along with his name and address cards. These are run through the computer which, under the control of a special program, calculates the quality points and quality point ratio. This program has an elaborate system of checks built into it to see that all the cards are present, that each has a proper grade on it, etc. The computer re-punches the Grade Cards with the proper quality points on it. Credit hours are corrected in the case of a failure or withdrawal. The computer then punches the #4 card which gives the total hours and quality points for the semester, and the #5 card which gives the cumulative hours and quality points, adjusting for repeated courses when necessary. Finally, the #6 card is punched giving the student's current quality point ratio along with any appropriate action such as "Dean's List," "Probation Removal," etc. These are then printed on the student's report card.

Four copies of the report card are made: one for the student, one for his parents, one for his high school, and one is kept in the Registrar's office. Special copies of the reports for students in each major field are printed and given to the head of each department. Copies of Boarders' marks are given to the dorm prefects, and labels are printed for attaching to the student's permanent record.

The computer has kept track of the total number of grades for each department, course and teacher. These are printed and dittoed, and distributed to appropriate offices and teachers. From the #6 cards, Probation Removal list, Probation List, and Probation Failure List are prepared, as well as the Dean's List.

V. BILLING PROCEDURE

The same basic set of cards is used in preparing each student's bill. A special program has been written, which, from the coded information on the student's name card, causes the computer to punch out special cards for the various charges such as tuition, room and board, matriculation and graduation fees, etc. These are then sorted with the students Grade Cards which contain any appropriate lab fees. Special students are charged by the semester hours, and the computer program calculates the tuition for those special cases. The computer keeps totals for each of the various charges, and for the lab fees by department, so that the business office has a check on the amount and distribution of the incoming funds.

Any credit that the student has with the business office by means of previous payments is entered on the bill by hand. A copy of the bill is sent to the student's parents, and a copy kept in the business office.

VI. CONCLUSION

We are very pleased with the system just described. The 1620 is not exactly a data processing machine nor do we have all the equipment that we might have. We do not feel that our small enrollment justifies a larger data processing machine, but we do feel that we are making very good use of the equipment that we do have on hand. Although it still requires a large amount of hand work, it is not nearly as much as we did previously, and the amount of time spent on the various records has been cut quite drastically.

MATRIX STRUCTURAL ANALYSIS

Everett L. Cook

Wichita State University
Wichita, Kansas

The program described herein may be used to analyze structures by either the force method or the displacement method. The method of analysis is specified by a control card at execution time. The input data is, of course, a function of the method to be used.

The program has two unique features:

1. A recursive technique is used to solve for the redundants or kinematic deficiencies rather than a direct matrix inversion.
2. The flexibility or stiffness matrix of the unassembled structure is stored in a special compressed format.

The program presented here was compiled in PDQ Fortran and run on a 40k 1620 with card input-output. However, the original version was a four-part program for a 20k 1620 (Ref. 2).

Matrix Force/Displacement Structural Analysis

The analogy between the matrix force and matrix displacement methods of structural analyses was first tabulated by Argyris and Kelsey (1). Pestel and Leckie (3) have pursued this analogy and have developed an excellent notation which will be used here.

The analogy is based on the inverse relationship of the following basic parameters:

External Forces = $\{f\}$ $\leftrightarrow\!\!\rightarrow$ $\{d\}$ = External Displacements

Internal Forces = $\{p\}$ $\leftrightarrow\!\!\rightarrow$ $\{v\}$ = Internal Displacements

Element Flexibility = $[F]$ $\leftrightarrow\!\!\rightarrow$ $[K]$ = Element Stiffness

Sponsored by the U. S. Army Research Office - Durham under Grant No. DA-ARO(D)-31-124-C482.

Since the force method is easier to explain, the basic equations for this method will be stated and explained, and then the analogy with the displacement method will be shown. The two equations of final interest in either analysis are

$$\{d\} = [F_d]\{f\} \tag{1}$$

and

$$\{p\} = [B]\{f\}, \tag{2}$$

since Equation (1) expresses the external displacements in terms of the known external forces and Equation (2) gives the internal forces in terms of these same external forces.

The matrix equations which must be solved in order to obtain the two rectangular matrices, $[F_d]$ and $[B]$, are as follows:

$$[D_{10}] = [B_1]^T [F_v] [B_0] \tag{3}$$

$$[D_{11}] = [B_1]^T [F_v] [B_1] \tag{4}$$

$$[X] = -[D_{11}]^{-1} [D_{10}] \tag{5}$$

$$[B] = [B_0] + [B_1] [X] \tag{6}$$

$$[F_d] = [B]^T [F_v] [B] \tag{7}$$

Three of the matrices in the above equations must be formulated from the physical and geometric properties of the structure:

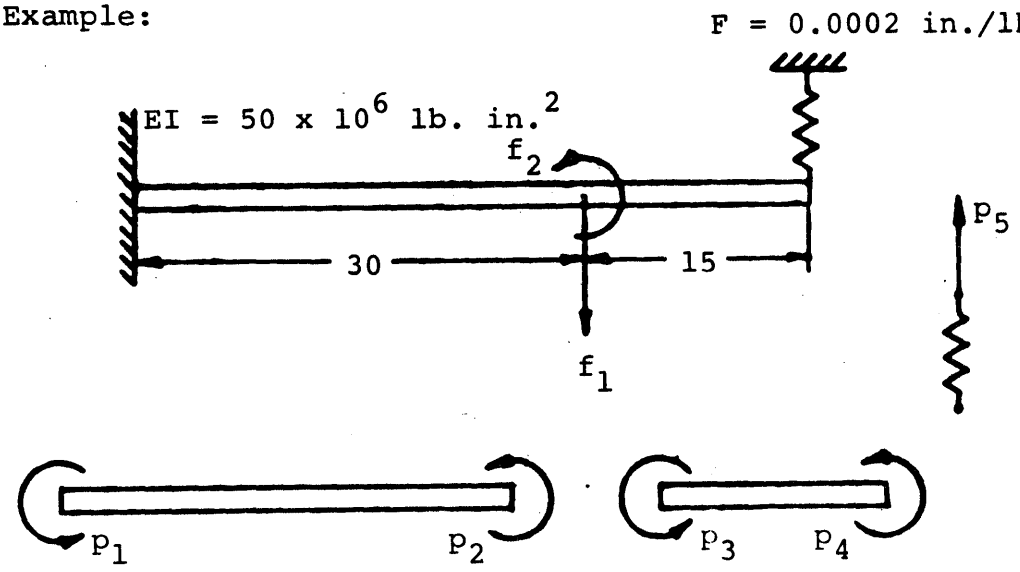
$[B_0]$ = a set of internal forces in the structure due to unit values of the external forces. Each column represents a set of internal forces due to one of the external loads.

$[B_1]$ = a set of internal forces in the structure due to unit values of the redundants. Each column represents a set of internal forces due to one of the redundants. For a statically determinate structure $[B_1] = [0]$ and $[B] = [B_0]$.

$[F_v]$ = the flexibility matrix of the unassembled structure. This is a diagonally partitioned matrix where the submatrices are the element flexibility matrices.

These matrices are the input required for a matrix force analysis utilizing the program being discussed.

Example:



Assume that the force in the spring, p_5, is the redundant.

$$[B_0] = \begin{array}{c} \begin{matrix} f_1 & f_2 \end{matrix} \\ \begin{bmatrix} 30.0 & -1.0 \\ 0.0 & 1.0 \\ 0.0 & 0.0 \\ 0.0 & 0.0 \\ 0.0 & 0.0 \end{bmatrix} \end{array} \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{matrix} \qquad [B_1] = \begin{array}{c} \begin{matrix} x_1 \end{matrix} \\ \begin{bmatrix} -45.0 \\ 15.0 \\ -15.0 \\ 0.0 \\ 1.0 \end{bmatrix} \end{array} \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{matrix}$$

$$[F_v] = 10^{-6} \begin{bmatrix} 0.20 & -0.10 & & & \\ -0.10 & 0.20 & & & \\ & & 0.10 & -0.05 & \\ & & -0.05 & 0.10 & \\ & & & & 200. \end{bmatrix} \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{matrix}$$

The matrix displacement equations equivalent to Equations (3) - (6) are

$$[C_{10}] = [A_1]^T[K_p][A_0] \tag{8}$$

$$[C_{11}] = [A_1]^T[K_p][A_1] \tag{9}$$

$$[Y] = -[C_{11}]^{-1}[C_{10}] \tag{10}$$

$$[A] = [A_0] + [A_1][Y] \tag{11}$$

$$[K_f] = [A]^T[F_v][A] \tag{12}$$

Two additional equations are required in order to obtain the external displacements and the internal forces with Equations (1) and (2).

$$[F_d] = [K_f]^{-1} \tag{13}$$

$$[B] = [K_p][A][K_f]^{-1} \tag{14}$$

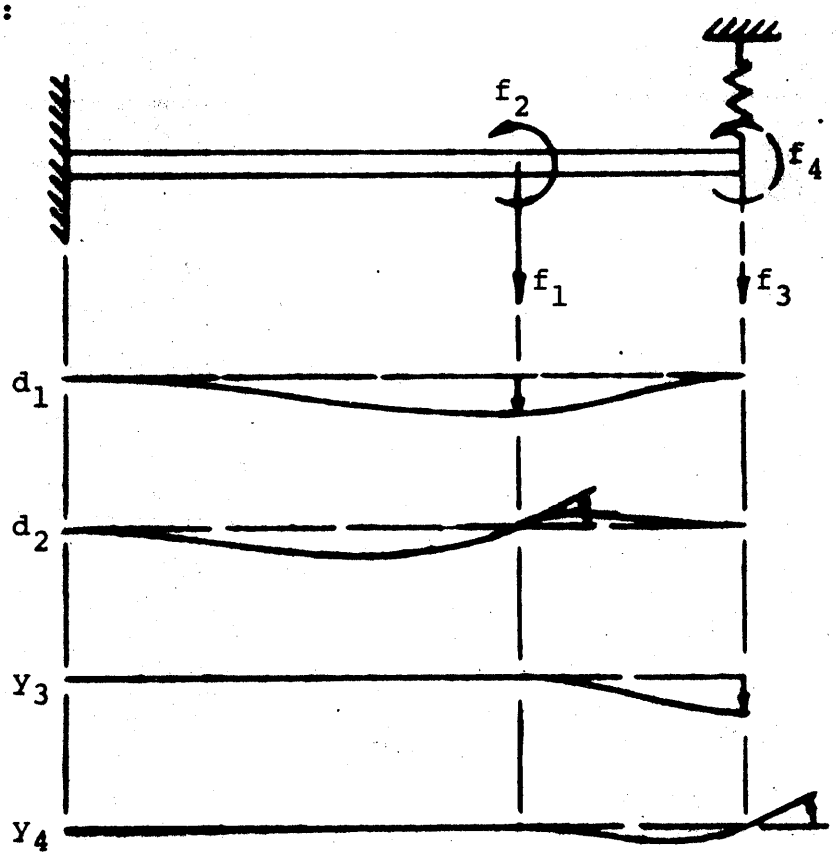
The three input matrices required for a displacement analysis are

$[A_0]$ = the internal displacements due to unit displacements at the external loads.

$[A_1]$ = the internal displacements due to unit displacements at the kinematic deficiencies. A kinematic deficiency exists for each degree of freedom for which there is no external load. If there are loads corresponding to all unconstrained external displacements, $[A_1] = [0]$ and $[A] = [A_0]$.

$[K_p]$ = the stiffness matrix for the unassembled structure.

Example:



$$[A_0] = \begin{array}{c} \begin{matrix} d_1 & d_2 \end{matrix} \\ \begin{bmatrix} 0.033333334 & 0.0 \\ 0.033333334 & 1.0 \\ -0.066666667 & 1.0 \\ -0.066666667 & 0.0 \\ 0.0 & 0.0 \end{bmatrix} \end{array} \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{matrix}$$

$$[A_1] = \begin{array}{c} \begin{matrix} y_3 & y_4 \end{matrix} \\ \begin{bmatrix} 0.0 & 0.0 \\ 0.0 & 0.0 \\ 0.066666667 & 0.0 \\ 0.066666667 & 1.0 \\ 1.0 & 0.0 \end{bmatrix} \end{array} \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{matrix}$$

$$[K_p] = 10^6 \begin{bmatrix} 6.6666666 & 3.3333333 & & & \\ 3.3333333 & 6.6666666 & & & \\ & & 13.333333 & 6.6666666 & \\ & & 6.6666666 & 13.333333 & \\ & & & & 0.005 \end{bmatrix} \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{matrix}$$

Recursion Analysis

The direct approach to the solution of the matrix force or displacement equations requires the inversion of either $[D_{11}]$ or $[C_{11}]$. Since the order of $[D_{11}]$ is equal to the number of redundants and the order of $[C_{11}]$ is equal to the number of kinematic deficiencies, the required inversion may be relatively large. Pestel and Leckie (3) have suggested a recursive technique whereby the redundants, or deficiencies, are divided into a number of subgroups, and an equal number of lower-ordered inversions are performed. In attempting to program this recursive technique, it was discovered that, not only is the programming simplified, the data storage requirements are minimized if the redundants, or deficiencies, are considered one at a time. Therefore, if there are n redundants, or deficiencies, there are n one-by-one inversions or, in fact, no inversions at all. It should be noted that a displacement analysis does require an inversion; however, the maximum order is five for the present program.

The basic recursive equations for the force method are as follows:

$$\{D_{01}^{i}\} = \{B_{1}^{i}\}^{T}[F_{v}][B_{01}^{i}] \tag{15}$$

$$\{X^{i}\} = -\{D_{0}^{i}\}/D_{1}^{i} \tag{16}$$

$$[B_{01}^{i+1}] = [B_{0}^{i}] + \{B_{1}^{i}\}\{X^{i}\}^{T} \tag{17}$$

Equations (3) and (4) have been combined to obtain Equation (15). These equations could be written

$$[D_{10}|D_{11}] = [B_{1}]^{T}[F_{v}][B_{0}|B_{1}] \tag{18}$$

For the first recursion, only the last column of the matrix $[B_0|B_1]$ is taken as the effective $\{B_1^1\}$. As a result,

both $\{D_{01}^1\}$ and $\{X^1\}$ are column vectors rather than rectangular matrices. The divisor in Equation (16) is the last element in $\{D_{01}^i\}$ and the redundants are actually computed and stored in $\{D_{01}^i\}$. The $[B_{01}^{i+1}]$ calculated with Equation (17) has one less column than $[B_{01}^i]$ and, after n recursions, $[B_{01}^{n+1}]$ is the unit internal force matrix [B].

The recursion equations for the matrix displacement method can be written directly from the analogy as

$$\{C_{01}\} = \{A_1^i\}^T[K_p][A_{01}^i] \tag{19}$$

$$\{Y^i\} = -\{C_0^i\}/C_1^i \tag{20}$$

$$\{A_{01}^{i+1}\} = [A_0^i] + \{A_1^i\}\{Y^i\}^T \tag{21}$$

Storage of $[F_v]$ or $[K_p]$

One of the major problems encountered in matrix structural analyses of the type being described here is the large size of the stiffness or flexibility matrix for the unassembled structure. However, since these matrices are diagonally partitioned, it is possible to store them in a compressed format. The present program uses an array with the number of rows equal to the maximum number of internal forces (50) and five columns. The number of columns restricts the size of the element flexibility, or stiffness, matrix which can be accommodated. Five columns were chosen in order to be able to analyze structures which contain rectangular or trapezoidal plate elements.

The element stiffness, or flexibility, matrices are right-justified in the array and the routine for calculating the product $\{A_1^i\}^T[K_p]$, or $\{B_1^i\}^T[F_v]$, determines the element size by searching a row until it finds a nonzero coefficient. Since the premultiplication matrix is a row matrix, the multiplication is reasonably simple.

Results for Example

FORCE METHOD

INTERNAL FORCES

+.12445820E+02	+.30960000E-02
+.58513932E+01	+.66563470E+00
-.58513932E+01	+.33436533E+00
+.00000000E-50	+.00000000E-50
+.39009288E+00	-.22291022E-01

DISPLACEMENTS

+.57120742E-04	-.19783282E-05
-.19783281E-05	+.19876162E-06

DISPLACEMENT METHOD

INTERNAL FORCES

+.12445820E+02	+.30959600E-02
+.58513930E+01	+.66563470E+00
-.58513927E+01	+.33436528E+00
+.57176470E-06	-.87529410E-07
+.39009287E+00	-.22291022E-01

DISPLACEMENTS

+.57120743E-04	-.19783282E-05
-.19783281E-05	+.19876161E-06

References

(1) Argyris, J. H., and S. Kelsey. Energy Theorems and Structural Analysis. Butterworths, London, 1960.

(2) Lambert, G. E. "A Digital Computer Program Utilizing the Matrix Force and Matrix Displacement Methods of Structural Analysis." (unpublished M.S. thesis, Wichita State University, 1965).

(3) Pestel, E. C., and F. A. Leckie. Matrix Methods in Elastomechanics. McGraw-Hill, New York, 1963.

MATRIX FORCE/DISPLACEMENT PROGRAM

E.L. COOK AND G.E. LAMBERT

REVISED MARCH 1966

THIS PROGRAM MAY BE USED TO ANALYZE STRUCTUTES BY EITHER THE MATRIX FORCE METHOD OR THE MATRIX DISPLACEMENT METHOD. THE STRUCTURE MAY HAVE ANY COMBINATION OF DIFFERENT TYPES OF ELEMENTS AS LONG AS NO ELEMENT HAS A FLEXIBILITY OR STIFFNESS MATRIX LARGER THAN FIVE-BY-FIVE. OTHER RESTRICTIONS ARE-

1. MAXIMUM NUMBER OF ELEMENTS = 50
2. MAXIMUM NUMBER OF INTERNAL FORCES = 50
3. MAXIMUM NUMBER OF EXTERNAL FORCES = 5
4. MAXIMUM NUMBER OF EXTERNAL FORCES PLUS REDUNDANTS OR KINEMATIC DEFICIENCIES = 28

THE INPUTS TO THE PROGRAM ARE AS FOLLOWS-

CARD 1	PARAMETER CARD - NUMBER OF ELEMENTS (K), NUMBER OF INTERNAL FORCES (L), NUMBER OF EXTERNAL FORCES (M), NUMBER OF REDUNDANTS OR DEFICIENCIES (N), NUMBER OF NONZERO ELEMENTS IN B0 OR A0 (NZEB0), NUMBER OF NONZERO ELEMENTS IN B1 OR A1 (NZEB1), AND A DESIGNATION FOR THE METHOD BEING USED (KK). FOR THE FORCE METHOD, KK=1, AND FOR THE DISPLACEMENT METHOD, KK=2. THESE PARAMETERS MUST BE IN FIXED POINT NOTATION (NO DECIMAL POINTS) AND MUST BE SEPARATED BY AT LEAST ONE BLANK. I.E.,

12 24 2 11 7 54 1

CARDS 2 THRU L+1	ELEMENT FLEXIBILITIES OR STIFFNESSES - THESE CARDS CONTAIN THE ELEMENT FLEXIBILITIES OR STIFFNESSES IN A SPECIAL FORMAT. EACH CARD MUST CONTAIN FIVE NUMBERS IN FLOATING POINT NOTATION (DECIMAL POINTS REQUIRED). A TWO-FORCE MEMBER WILL HAVE A SINGLE CARD WITH FOUR ZEROS FOLLOWED BY THE FLEXIBILITY OR STIFFNESS OF THE ELEMENT-

0.0 0.0 0.0 0.0 .66666667E-06

A BEAM ELEMENT(SAY PINNED AT THE ENDS WITH MOMENTS AT BOTH ENDS) WILL HAVE TWO CARDS WITH THREE ZEROS FOLLOWED BY THE ELEMENT FLEXIBILITY OR STIFFNESS. I.E.,

0.0 0.0 0.0 .11111111E-06 -.05555555E-06
0.0 0.0 0.0 -.05555555E-06 .11111111E-06

AND SO ON UP TO A PLATE ELEMENT WHICH WILL HAVE FIVE CARDS CONTAINING AN ELEMENT FLEXIBILITY OR STIFFNESS MATRIX OF THE FORM -

.40000E-05 .29999E-05 .99999E-06 -.33333E-06 -.33333E-06
.29999E-05 .86666E-05 .56666E-05 .29999E-05 -.29999E-05
.99999E-06 .56666E-05 .66666E-05 .26666E-05 -.33333E-05
-.33333E-06 .29999E-05 .26666E-05 .39999E-05 -.19999E-05
-.33333E-06 -.29999E-05 -.33333E-05 -.19999E-05 .39999E-05

NEXT NZEB0 CARDS	NONZERO ELEMENTS IN B0 OR A0 - EACH OF THESE CARDS CONTAINS ONE NONZERO ELEMENT OF B0 OR A0 PRECEEDED BY NUMBERS INDICATING THE ROW AND COLUMN IN WHICH IT APPEARS. THE ROW AND COLUMN ARE IN FIXED POINT NOTATION AND THE ELEMENT IS IN FLOATING POINT NOTATION, I.E.,

10 3 -.70710680E+01

NEXT NZEB1 CARDS	NONZERO ELEMENTS IN B1 OR A1 - THESE CARDS ARE REQUIRED ONLY FOR STATICALLY INDETERMINATE OR KINEMATICALLY DEFICIENT STRUCTURES. THEY CONTAIN THE NONZERO ELEMENTS OF B1 OR A1 IN THE SAME FORMAT AS THE CARDS IN THE PRECEDING GROUP.

GENERAL OPERATING INSTRUCTIONS

TO LOAD A NEW DECK INTO THE COMPUTER, PLACE THE DECK IN THE 1622 READ HOPPER, DEPRESS INSTANT STOP AND RESET ON THE 1620, AND DEPRESS LOAD(NOT READER START) ON THE 1622.

THE 1622 ALWAYS STOPS PRIOR TO READING THE LAST TWO CARDS OF A DECK. DEPRESS READER START ON THE 1622 TO READ THESE LAST TWO CARDS.

TO PUNCH CARDS, BLANK CARDS MUST BE PLACED IN THE PUNCH HOPPER OF THE 1622 AND PUNCH START MUST BE DEPRESSED. WHEN THE 1622 STOPS PUNCHING, THERE ARE STILL THREE CARDS IN THE MACHINE(ONE PUNCHED AND TWO BLANK). TO GET THESE CARDS, EMPTY THE PUNCH HOPPER AND DEPRESS NONPROCESS RUNOUT ON THE PUNCH END OF THE 1622.

TO STOP EXECUTION OF A PROGRAM, DEPRESS INSTANT STOP ON THE 1620.

TO RESTART A PROGRAM, DEPRESS RESET, INSERT, RELEASE, AND START ON THE 1620.

TO EMPTY THE READ UNIT, REMOVE ALL CARDS FROM THE READ HOPPER, DEPRESS READER STOP ON THE 1622, AND HOLD THE 1622 READER NONPROCESS RUNOUT SWITCH IN THE ON POSITION UNTIL TWO CARDS FALL INTO THE REJECT STACKER.

STEP-BY-STEP PROCEDURE

1. ON THE 1620 CONSOLE, SET THE PARITY AND I/O SWITCHES TO STOP AND THE OFLOW SWITCH TO PROGRAM. SET PROGRAM SWITCHES 1, 2, 3, AND 4 TO OFF FOR NORMAL OPERATION. IF SWITCH 4 IS SET TO ON, A PROGRAM TRACE WILL BE PUT INTO OPERATION.

2. PLACE THE PROGRAM DECK IN THE READ HOPPER OF THE 1622.

3. PLACE THE INPUT DATA ON TOP OF THE PROGRAM DECK AND LOAD THESE TWO DECKS(SEE THE GENERAL OPERATING INSTRUCTIONS). AFTER THE PROGRAM DECK HAS BEEN READ, THE DATA WILL BE READ AND PUNCHED. THE PROGRAM MAY TAKE SEVERAL MINUTES FOR LARGE STRUCTURES. THEREFORE, THE PROGRAM SHOULD NOT BE STOPPED UNLESS IT IS OBVIOUS THAT IT IS TAKING TOO MUCH TIME.

4. AFTER THE INTERNAL FORCES AND DISPLACEMENTS HAVE BEEN CALCULATED AND PUNCHED, THE MESSAGE - EMPTY PUNCH AND LOAD DATA - WILL BE TYPED. REMOVE THE PUNCHED OUTPUT FROM THE PUNCH STACKER AFTER THE PUNCH UNIT HAS BEEN EMPTIED.

5. LIST THE OUTPUT ON THE 407 ACCOUNTING MACHINE.

6. IT IS NOT NECESSARY TO RELOAD THE PROGRAM DECK IN ORDED TO RUN ANOTHE SET OF DATA. PLACE THE NEW DATA IN THE 1622 READ HOPPER AND DEPRESS READER START(NOT LOAD) AND PUNCH START ON THE 1622.

```
C  C  MATRIX FORCE/DISPLACEMENT PROGRAM
C
C     E.L. COOK AND G.E. LAMBERT - REVISED MARCH 1966
C
      DIMENSION FV(50,5),B01(50,28),B1TFV(50),D01(28),FD(5,5)
C
C     PART I - INPUT
C
      BEGIN TRACE
C
C     READ AND PUNCH PARAMETERS
    1 READ 100,K,L,M,N,NZEB0,NZEB1,KK
      GO TO(2,3),KK
    2 PUNCH 301
      GO TO 4
    3 PUNCH 302
    4 PUNCH 300,K,L,M,N,NZEB0,NZEB1
C
C     READ AND PUNCH FV OR KP BY ROWS, FIVE VALUES PER CARD
      GO TO(5,6),KK
    5 PUNCH 311
      GO TO 7
    6 PUNCH 312
    7 DO 8 I=1,L
      READ 101,FV(I,1),FV(I,2),FV(I,3),FV(I,4),FV(I,5)
    8 PUNCH 101,FV(I,1),FV(I,2),FV(I,3),FV(I,4),FV(I,5)
C
C     CLEAR B01 OR A01
      MN=M+N
      DO 11 I=1,L
      DO 11 J=1,MN
   11 B01(I,J)=0.0
C
C     READ AND PUNCH NONZERO ELEMENTS IN B0 OR A0
      GO TO(12,13),KK
   12 PUNCH 321
      GO TO 14
   13 PUNCH 322
   14 DO 15 IA=1,NZEB0
      READ 102,I,J,B01(I,J)
   15 PUNCH 102,I,J,B01(I,J)
      IF(N)16,31,17
   16 PRINT 333
      STOP
C
C     READ AND PUNCH NONZERO ELEMENTS IN B1 OR A1
   17 GO TO(18,19),KK
   18 PUNCH 331
      GO TO 20
   19 PUNCH 332
   20 DO 21 IA=1,NZEB1
      READ 102,I,J,B1IJ
      PUNCH 102,I,J,B1IJ
      JM=J+M
   21 B01(I,JM)=B1IJ
```

```
C  C  PART II - RECURSION ANALYSIS

   31 DO 49 IR=1,N
C     B1 TRANSPOSE TIMES FV OR A1 TRANSPOSE TIMES KP
      DO 32 I=1,L
   32 B1TFV(I)=0.0
      J=1
      DO 45 IS=1,K
      IF(FV(J,1))50,35,33
   33 J4=J+4
      DO 34 IA=1,5
      IAJ=IA+J-1
      DO 34 JA=J,J4
   34 B1TFV(IAJ)=B1TFV(IAJ)+B01(JA,MN)*FV(JA,IA)
      J=J+5
      GO TO 45
   35 IF(FV(J,2))50,38,36
   36 J3=J+3
      DO 37 IA=2,5
      IAJ=IA+J-2
      DO 37 JA=J,J3
   37 B1TFV(IAJ)=B1TFV(IAJ)+B01(JA,MN)*FV(JA,IA)
      J=J+4
      GO TO 45
   38 IF(FV(J,3))50,41,39
   39 J2=J+2
      DO 40 IA=3,5
      IAJ=IA+J-3
      DO 40 JA=J,J2
   40 B1TFV(IAJ)=B1TFV(IAJ)+B01(JA,MN)*FV(JA,IA)
      J=J+3
      GO TO 45
   41 IF(FV(J,4))50,44,42
   42 J1=J+1
      DO 43 IA=4,5
      IAJ=IA+J-4
      DO 43 JA=J,J1
   43 B1TFV(IAJ)=B1TFV(IAJ)+B01(JA,MN)*FV(JA,IA)
      J=J+2
      GO TO 45
   44 B1TFV(J)=B01(J,MN)*FV(J,5)
      J=J+1
   45 CONTINUE
C     CALCULATION OF D01(B1TFV TIMES B01 OR A1TKP TIMES A01)
      DO 46 I=1,MN
      D01(I)=0.0
      DO 46 J=1,L
   46 D01(I)=D01(I)+B1TFV(J)*B01(J,I)
C     CALCULATION OF REDUNDANTS OR DEFICIENCIES(STORED IN D01)
      DO 47 I=1,MN
   47 D01(I)=-D01(I)/D01(MN)
C     CALCULATION OF NEW B01 OR A01
      DO 48 I=1,L
      DO 48 J=1,MN
   48 B01(I,J)=B01(I,J)+B01(I,MN)*D01(J)
   49 MN=MN-1
      GO TO 71
   50 PRINT 334
      STOP
```

041

```
C  C  PART III - CALCULATION OF FLEXIBILITY OR STIFFNESS MATRIX

C     CALCULATION OR FV TIMES B OR KP TIMES A(STORED IN B)
   71 M1=M+1
      MM=M+M
      DO 72 I=1,L
      DO 72 J=M1,MM
   72 B01(I,J)=0.0
      DO 85 IR=1,M
      IRM=IR+M
      J=1
      DO 85 IS=1,K
      IF(FV(J,1))50,75,73
   73 J4=J+4
      DO 74 IA=1,5
      IAJ=IA+J-1
      DO 74 JA=J,J4
   74 B01(IAJ,IRM)=B01(IAJ,IRM)+B01(JA,IR)*FV(JA,IA)
      J=J+5
      GO TO 85
   75 IF(FV(J,2))50,78,76
   76 J4=J+3
      DO 77 IA=2,5
      IAJ=IA+J-2
      DO 77 JA=J,J4
   77 B01(IAJ,IRM)=B01(IAJ,IRM)+B01(JA,IR)*FV(JA,IA)
      J=J+4
      GO TO 85
   78 IF(FV(J,3))50,81,79
   79 J4=J+2
      DO 80 IA=3,5
      IAJ=IA+J-3
      DO 80 JA=J,J4
   80 B01(IAJ,IRM)=B01(IAJ,IRM)+B01(JA,IR)*FV(JA,IA)
      J=J+3
      GO TO 85
   81 IF(FV(J,4))50,84,82
   82 J4=J+1
      DO 83 IA=4,5
      IAJ=IA+J-4
      DO 83 JA=J,J4
   83 B01(IAJ,IRM)=B01(IAJ,IRM)+B01(JA,IR)*FV(JA,IA)
      J=J+2
      GO TO 85
   84 B01(J,IRM)=B01(J,IR)*FV(J,5)
      J=J+1
   85 CONTINUE
C
C     CALCULATION OF FD OR KP
      DO 88 I=1,M
      IM=I+M
      DO 88 J=1,M
      FD(I,J)=0.0
      DO 88 K=1,L
      FD(I,J)=FD(I,J)+B01(K,IM)*B01(K,J)
      IF(ABS(FD(I,J))-.10**20)87,87,88
   87 FD(I,J)=0.0
   88 CONTINUE
```

```
C  C  PART IV - INTERNAL FORCES AND DISPLACEMENTS - DISPLACEMENT METHOD

      GO TO(98,600),KK
C
C     INVERSION OF KF TO OBTAIN FD
  600 DO 604 I=1,M
      STORE=FD(I,I)
      FD(I,I)=1.0
      DO 601 J=1,M
  601 FD(I,J)=FD(I,J)/STORE
      DO 604 K=1,M
      IF(K-I)602,604,602
  602 STORE=FD(K,I)
      FD(K,I)=0.0
      DO 603 J=1,M
  603 FD(K,J)=FD(K,J)-STORE*FD(I,J)
  604 CONTINUE
C
C     CALCULATION OF B
      DO 99 I=1,L
      DO 99 J=1,M
      B01(I,J)=0.0
      DO 99 K=1,M
      IM=K+M
   99 B01(I,J)=B01(I,J)+B01(I,IM)*FD(K,J)
C
C     FORMAT STATEMENTS
C
  100 FORMAT(715)
  101 FORMAT(5E16.8)
  102 FORMAT(2I5,E16.8)
  300 FORMAT(615/)
  301 FORMAT(26H FORCE METHOD - PARAMETERS/)
  302 FORMAT(33H DISPLACEMENT METHOD   PARAMETERS/)
  311 FORMAT(22H ELEMENT FLEXIBILITIES/)
  312 FORMAT(20H ELEMENT STIFFNESSES/)
  321 FORMAT(/23H NONZERO ELEMENTS IN B0/)
  322 FORMAT(/23H NONZERO ELEMENTS IN A0/)
  331 FORMAT(/23H NONZERO ELEMENTS IN B1/)
  332 FORMAT(/23H NONZERO ELEMENTS IN A1/)
  333 FORMAT(14H N IS NEGATIVE)
  334 FORMAT(29H NEGATIVE FIRST ELEMENT IN FV)
  335 FORMAT(26H EMPTY PUNCH AND LOAD DATA)
  400 FORMAT(5E16.8)
  401 FORMAT(/16H INTERNAL FORCES/)
  402 FORMAT(/14H DISPLACEMENTS/)
```

141

```
C  C  PART V - INTERNAL FORCES AND DEFLECTIONS - OUTPUT

C     INTERNAL FORCES
   98 PUNCH 401
      GO TO(611,621,631,641,651),M
  611 DO 612 I=1,L
  612 PUNCH 400,B01(I,1)
      GO TO 700
  621 DO 622 I=1,L
  622 PUNCH 400,B01(I,1),B01(I,2)
      GO TO 700
  631 DO 632 I=1,L
  632 PUNCH 400,B01(I,1),B01(I,2),B01(I,3)
      GO TO 700
  641 DO 642 I=1,L
  642 PUNCH 400,B01(I,1),B01(I,2),B01(I,3),B01(I,4)
      GO TO 700
  651 DO 652 I=1,L
  652 PUNCH 400,B01(I,1),B01(I,2),B01(I,3),B01(I,4),B01(I,5)
C
C     DISPLACEMENTS
  700 PUNCH 402
      GO TO(711,721,731,741,751),M
  711 DO 712 I=1,M
  712 PUNCH 400,FD(I,1)
      GO TO 800
  721 DO 722 I=1,M
  722 PUNCH 400,FD(I,1),FD(I,2)
      GO TO 800
  731 DO 732 I=1,M
  732 PUNCH 400,FD(I,1),FD(I,2),FD(I,3)
      GO TO 800
  741 DO 742 I=1,M
  742 PUNCH 400,FD(I,1),FD(I,2),FD(I,3),FD(I,4)
      GO TO 800
  751 DO 752 I=1,M
  752 PUNCH 400,FD(I,1),FD(I,2),FD(I,3),FD(I,4),FD(I,5)
C
  800 PRINT 335
      GO TO 1
C
      END TRACE
C
      END
```

DISPLACEMENT METHOD - PARAMETERS

+3 +5 +2 +2 +6 +4

ELEMENT STIFFNESSES

+.00000000E-50	+.00000000E-50	+.00000000E-50	+.66666666E+07	+.33333333E+07
+.00000000E-50	+.00000000E-50	+.00000000E-50	+.33333333E+07	+.66666666E+07
+.00000000E-50	+.00000000E-50	+.00000000E-50	+.13333333E+08	+.66666666E+07
+.00000000E-50	+.00000000E-50	+.00000000E-50	+.66666666E+07	+.13333333E+08
+.00000000E-50	+.00000000E-50	+.00000000E-50	+.00000000E-50	+.50000000E+04

NONZERO ELEMENTS IN A0

+1	+1	+.33333334E-01
+2	+1	+.33333334E-01
+2	+2	+.10000000E+01
+3	+2	+.10000000E+01
+3	+1	-.66666667E-01
+4	+1	-.66666667E-01

NONZERO ELEMENTS IN A1

+3	+1	+.66666667E-01
+4	+1	+.66666667E-01
+4	+2	+.10000000E+01
+5	+1	+.10000000E+01

INTERNAL FORCES

+.12445820E+02	+.30959600E-02
+.58513930E+01	+.66563470E+00
-.58513927E+01	+.33436528E+00
+.57176470E-06	-.87529410E-07
+.39009287E+00	-.22291022E-01

DISPLACEMENTS

+.57120743E-04	-.19783282E-05
-.19783281E-05	+.19876161E-06

FORCE METHOD - PARAMETERS

+3 +5 +2 +1 +3 +4

ELEMENT FLEXIBILITIES

+.00000000E-50	+.00000000E-50	+.00000000E-50	+.20000000E-06	-.10000000E-06
+.00000000E-50	+.00000000E-50	+.00000000E-50	-.10000000E-06	+.20000000E-06
+.00000000E-50	+.00000000E-50	+.00000000E-50	+.10000000E-06	-.50000000E-07
+.00000000E-50	+.00000000E-50	+.00000000E-50	-.50000000E-07	+.10000000E-06
+.00000000E-50	+.00000000E-50	+.00000000E-50	+.00000000E-50	+.20000000E-03

NONZERO ELEMENTS IN B0

+1	+1	+.30000000E+02
+1	+2	-.10000000E+01
+2	+2	+.10000000E+01

NONZERO ELEMENTS IN B1

+1	+1	-.45000000E+02
+2	+1	+.15000000E+02
+3	+1	-.15000000E+02
+5	+1	+.10000000E+01

INTERNAL FORCES

+.12445820E+02	+.30960000E-02
+.58513932E+01	+.66563470E+00
-.58513932E+01	+.33436533E+00
+.00000000E-50	+.00000000E-50
+.39009288E+00	-.22291022E-01

DISPLACEMENTS

+.57120742E-04	-.19783282E-05
-.19783281E-05	+.19876162E-06

COMPUTER DESIGN

OF

HEAVY MULTISTORY RIGID FRAME

INDUSTRIAL STRUCTURES

By
G.L. Kirby
N.M. Dudnikoff
V.L. Arndt

Presented at the
Western Region Summer COMMON Meeting
Denver, Colorado
July 7, 1966

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Introduction. -- During 1964, a Southwestern Copper Plant, undertook a major improvement and plant expansion program. Stearns-Roger Corporation, Engineers-Contractors, of Denver, Colorado designed and constructed the necessary facilities for this expansion.

The facilities consisted of:

1. Crushing plant.
2. Expansion of concentrator plant.
3. One new reverbatory furnace.
4. Associated conveyors and transfer towers.
5. Two new 10,000 ton live storage copper ore bins.
6. Modification of two existing 5,000 ton live storage bins.

Ore storage bin support design. -- This paper will cover the application of Stearns-Roger's Multistory Rigid Frame Program for analyzing and optimumizing the design of the two ore storage bin support frames. Figure 1 shows a typical bin with head house and support structure.

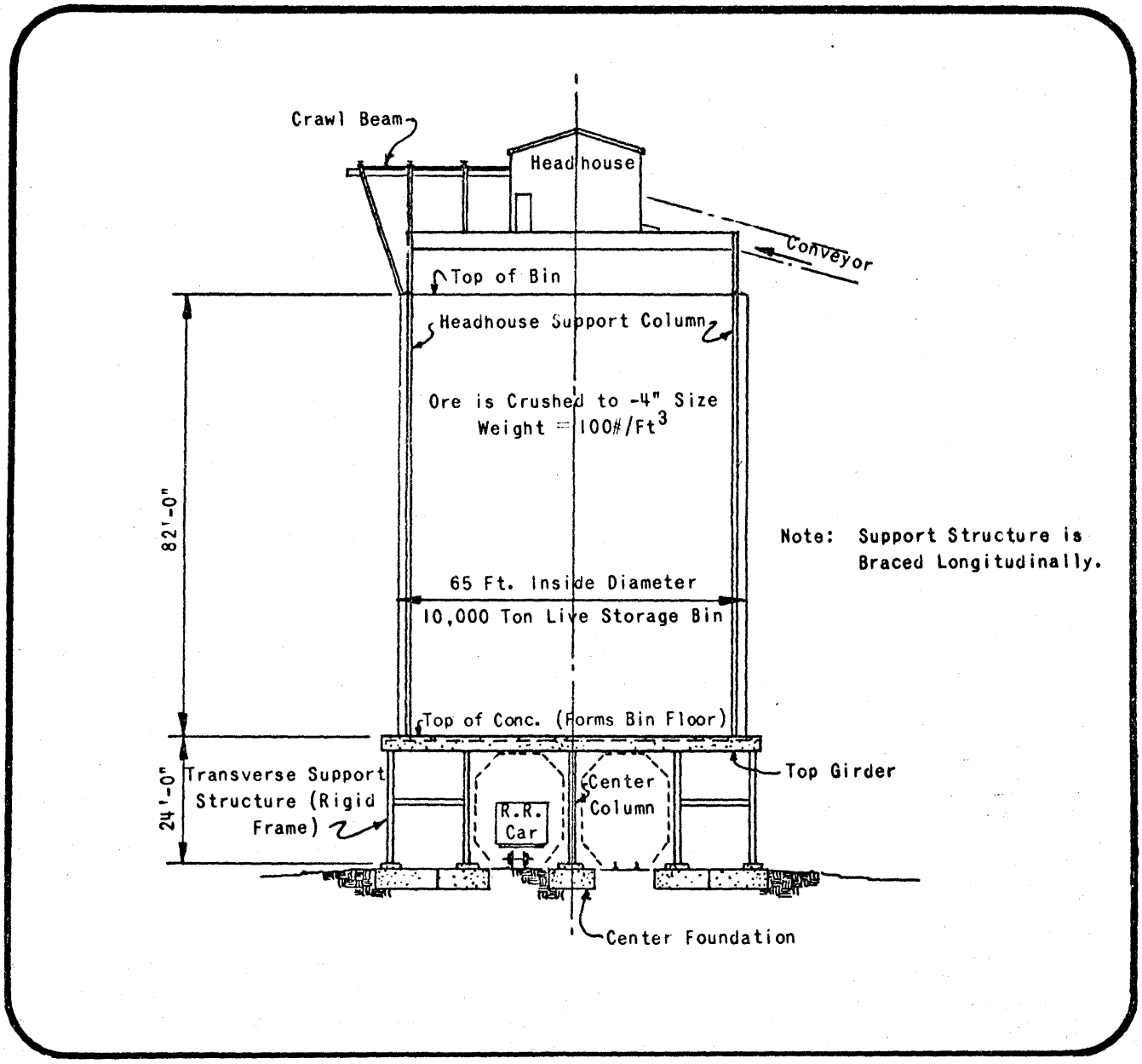


Figure 1. Elevation of 10,000 Ton Ore Storage Bin.

147

Design considerations. -- Reference should be made back to Figure 1 to fully comprehend the following design criteria:

1. Bin capacity - 10,000 ton established by customer.
2. Bottom elevation of bin established by both railroad clearance and arrangement of chutes used for loading railroad cars.
3. Bins were to be located between two existing bins as shown in Figure 2.

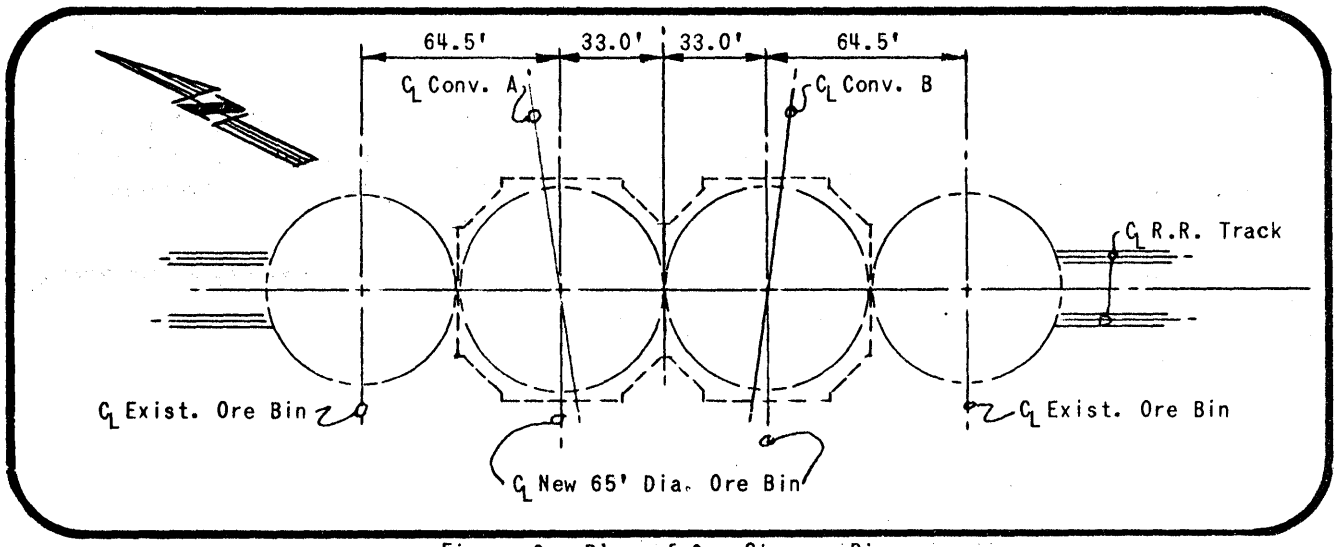


Figure 2. Plan of Ore Storage Bins.

4. Top elevation of bin established by 10,000 ton capacity, since all other dimensions were set.
5. Most column depths were limited by railroad clearance.
6. Railroad traffic had to be maintained to an ore concentrator located 7 miles away. It was mandatory that foundations, supports, and bins be constructed during normal rail traffic hauling 30,000 tons of ore per day (20 trains of 30 cars each).

Rigid frame supporting structure. -- A rigid frame design was selected because:

1. Flexibility provides better resistance to seismic activity (UBC-Zone1).
2. Flexibility provides better resistance to shock loading from blasting done inside the bins to break stoppages in plugged ore chutes.
3. Transverse bracing would limit accessibility to operating platforms located within the structure.
   A. Steel construction was selected to speed erection. A441 steel was utilized to reduce member (especially column) size and weight, thereby minimizing handling problems during erection.
   B. Due to the limited railroad clearance, pier size was kept to a minimum by using 5000 psi concrete.
4. The bin floor was constructed of cast in place concrete. The floor was connected to the top steel girder, thereby obtaining economy and rigidity from the composite construction. Floor beams were made continuous over supports where possible, for economical reasons, and also to decrease deflections.

I. Preliminary Design

Frame layout. -- Figure 3 shows the layout of the bin floor framing and the rigid frame supports. Frames A and C were analyzed on the IBM 1620 Computer. The results from these runs were ratioed to provide design moments and forces for other frames. The analysis and design of Frame C is covered in the remainder of this paper. Frame A was analyzed and designed similarly to Frame C.

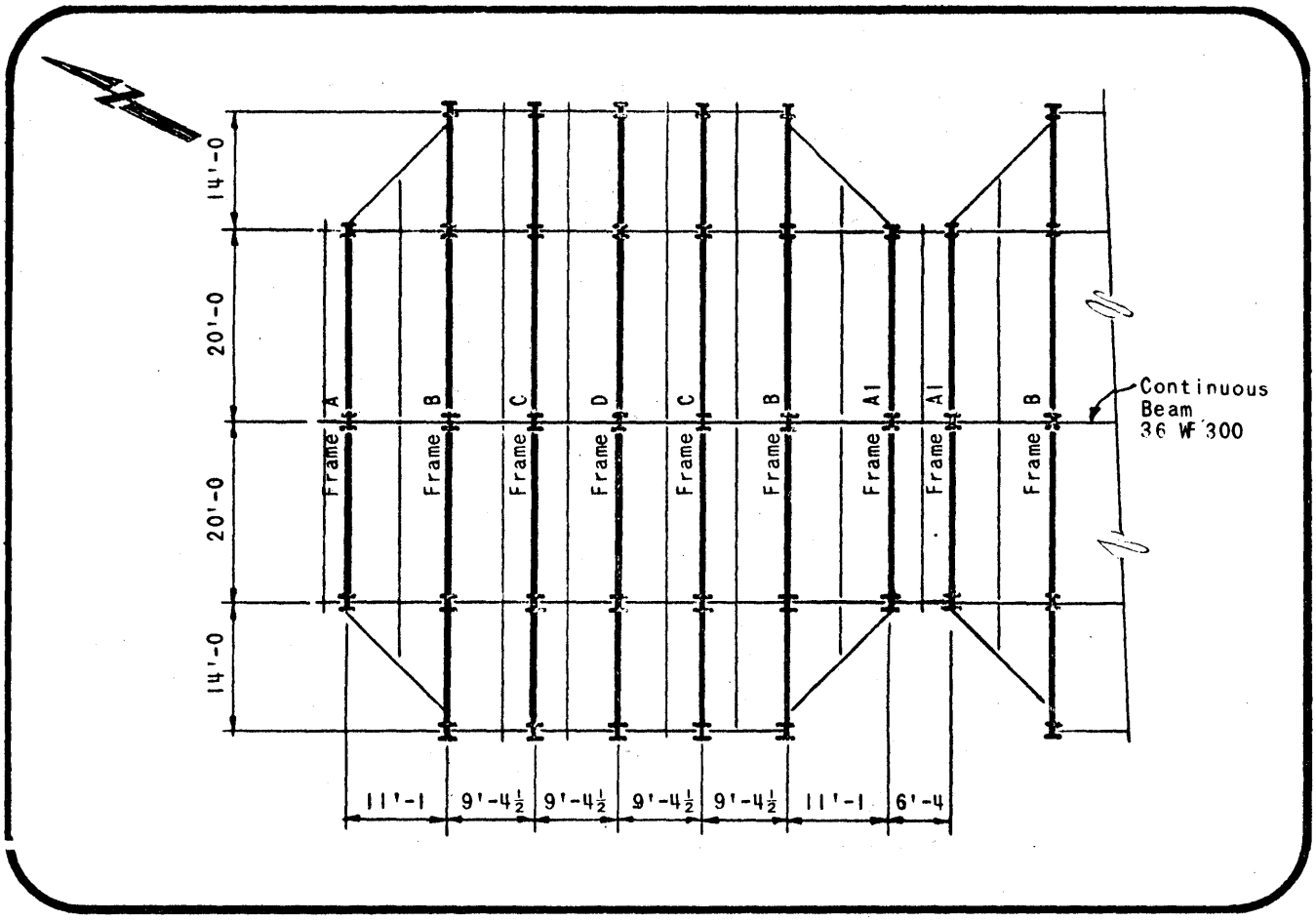


Figure 3. Plan Showing Rigid Frame Layout and Bin Floor Framing.

Loads. -- Vertical loads were distributed to the rigid frames from associated bin floor framing, and by assuming contributary widths over which distributed loads act. Dead and live loads amounted to almost 10 ksf. Zone 1 seismic and wind produced about the same lateral loading. A total lateral load of 230 kips was used for design. Foundations were located on rock with a 25 ksf allowable bearing pressure.

Center foundation size. -- Since railroad traffic could not be interrupted during construction, it was necessary to limit the size of the base pads for center foundations. It was also desirable to make the top girder continuous over the center column in order to minimize stresses and deflections in the bin floor. However, it was assumed that this configuration would produce greater loads on the center column than if the top girders were interrupted at this point. These two schemes were investigated on the computer to determine which design was more suitable, considering girder stress/rigidity vs foundation loads.

Scheme 1 - top girder pin-connected to column. -- Figure 4 shows the idealized frame which was analyzed on the computer. Loads, dimensions, and assumed moments of inertia are shown. At this stage of design, column bases were assumed pinned. Vertical and horizontal loads were assumed to be load conditions 1 and 2 respectively.

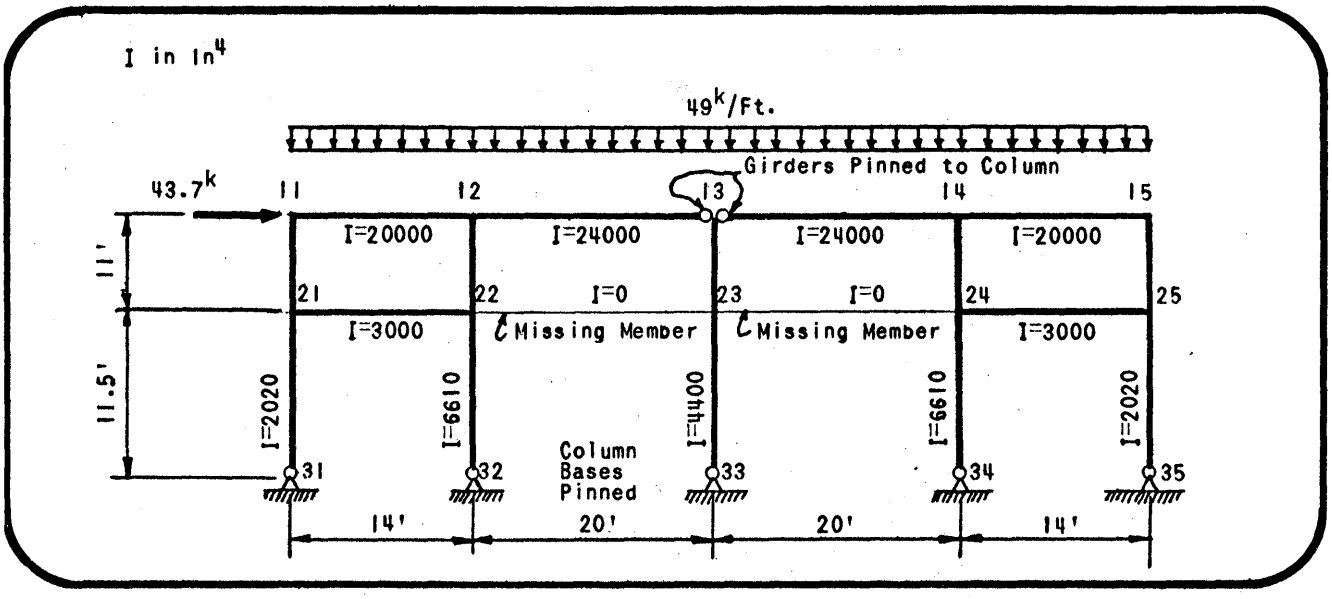


Figure 4. Frame and Loads - Scheme I.

Scheme I - data input sheets. -- The completed data sheets are given in Table I.

Table I (a) describes the structure.

Table I (b) gives structural properties - geometry, stiffness, connection rigidities, etc.

Table I (c) describes beam loads.

Table I (d) describes column loads.

A

Table I (a). Data Sheet for Support Structure - Frame Parameters.

IBM DATA FORMAT SHEET-MULTI-STORY RIGID FRAME

SHEET NO. 1 OF 5

CONTROL CARD

NO. OF LAST NODE		NO. OF BEAMS MISSING	NO. OF COLUMNS MISSING	NUMBER OF LOADS	MODULUS OF ELASTICITY (KIPS/IN^2)
I	J				
3	5	2		2	29000.

ORANGE CARD

1 5 10 15 20 25 30 35

151

1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80

STEARNS-ROGER CORPORATION MULTISTORY RIGID FRAME PROGRAM - G7

JOB NO. B- CUSTOMER- PROJECT

SUBJECT- BIN SUPPT. FRAME C- SCHEME I BY NMD DATE 5-6-64

NOTE:

1. Sheets A, B, C, D, Required For Complete Data Input.

FORM G7.1

5

B

Table I (b). Data Sheet for Support Structure - Geometrical Properties.

IBM DATA FORMAT SHEET-FOR FORMING STIFFNESS MATRIX

SHEET NO. 2 OF 5

JOB NO. B- DATE 5-6-64 BY NMD CH'K. VLA

CUSTOMER PROJECT

SUBJECT Bin Suppt. Frame C - Scheme I

Beams / Columns

I	J	MEMBER LENGTH L (ft.)	MOMENT OF INERTIA I (in^4)	RIGIDITY FACTOR R	FXD.END COL. BASE	C1	C2	C3
1	1	14.	20000.	100.				
1	2	20.	24000.	0.		3.	0.	0.
1	3	20.	24000.	0.		0.	0.	3.
1	4	14.	20000.	100.				
2	1	14.	3000.	100.				
2	2	20.	0.	100.				
2	3	20.	0.	100.				
2	4	14.	3000.	100.				
1	1	11.	2020.	100.				
1	2	11.	6610.	100.				
1	3	11.	4400.	100.				
1	4	11.	6610.	100.				
1	5	11.	2020.	100.				
2	1	11.5	2020.	100.				
2	2	11.5	6610.	100.				
2	3	11.5	4400.	100.				
2	4	11.5	6610.	100.				
2	5	11.5	2020.	100.				

I J MEMBER NO. | MEMBER LENGTH L (ft.) | MOMENT OF INERTIA I (in^4) | RIGIDITY FACTOR R | FXD.END COL. BASE | C-FACTORS NON-PRISMATIC MEMBERS: C1, C2, C3

FORM G7.2

152

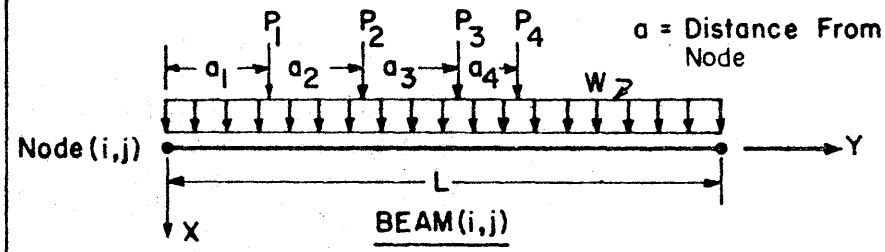
Table I (c). Data Sheet for Support Structure - Beam Loads.

C

IBM DATA FORMAT SHEET - BEAM DATA

SHEET NO. 3 OF 5

JOB NO. B- DATE 5-6-64 BY NMD CH'K. VLA



Notes

1. Forces Shown On The Beam Are In The Positive Direction. Enter Minus Values For Opposite Direction.
2. Prismatic Members With Loads Shown On This Sheet -Enter Data Indicated Above Divider Lines. Non-Prismatic Members Or For Loads Not Shown-Enter R=0 And Data Indicated Below Divider Lines.

I	J	BEAM NO.	LOAD COND. NO	BEAM LENGTH L(FT)	RIGIDITY FACTOR R	UNIF. LOAD W (K/FT)	CONCEN. LOAD P_1 (K)	POSITION a_1 (FT)	CONCEN. LOAD P_2 (K)	POSITION a_2 (FT)	CONCEN. LOAD P_3 (K)	POSITION a_3 (FT)	CONCEN. LOAD P_4 (K)	POSITION a_4 (FT)
										SUM SHEARS (K)	LEFT FIXED MOMENT FML (K-FT)	RIGHT FIXED MOMENT FMR (K-FT)	RIGHT FIXED SHEAR FQR (K)	SEE NOTE 2
1	1	1	1	14.	100.	4.9								
1	1	2	2	14.	100.									
1	2	1	1	20.	0.					980.	2450.	0.	-367.	
1	2	2	2	20.	0.									
1	3	1	1	20.	0.					980.	0.	-2450.	-613.	
1	3	2	2	20.	0.									
1	4	1	1	14.	100.	4.9								
1	4	2	2	14.	100.									
2	1	1	1	14.	100.									
2	1	2	2	14.	100.									
2	2	1	1	20.	100.									
2	2	2	2	20.	100.									
2	3	1	1	20.	100.									
2	3	2	2	20.	100.									
2	4	1	1	14.	100.									
2	4	2	2	14.	100.									

153

FORM G7.3

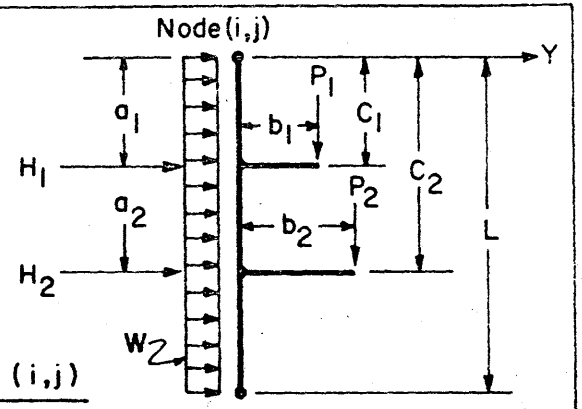
D

Table I (d). Data Sheet for Support Structure - Column Loads.

IBM DATA FORMAT SHEET — COLUMN DATA

JOB NO. B- SHEET NO. 4 OF 5

DATE 5-6-64 BY NMD CH'K. VLA



Notes

1. Forces Shown On The Column Are In The Positive Direction. Enter Minus Values For Opposite Direction.
2. Prismatic Member With Loads Shown On This Sheet-Enter Data Indicated Above Divider Lines. Non-Prismatic Members Or For Loads Not Shown—Enter R=0 And Data Indicated Below Divider Lines.

I	J	LOAD COND. NO	COLUMN LENGTH L(FT)	RIGIDITY FACTOR R	UNIF. LOAD W (K/FT)	CONCEN. LOAD H_1 (K)	POSITION a_1 (FT)	CONCEN. LOAD H_2 (K)	POSITION a_2 (FT)	CONCEN. LOAD P_1 (K)	POSIT. b_1 (FT)	POSITION C_1 (FT)	CONCEN LOAD P_2 (K)	POSIT. b_2 (FT)	POSITION C_2 (FT)
1	1	1	11.0	100.											
1	1	2	11.0	100.		43.7									
1	2	1	11.0	100.											
1	2	2	11.0	100.											
1	3	1	11.0	100.											
1	3	2	11.0	100.											
1	4	1	11.0	100.											
1	4	2	11.0	100.											
1	5	1	11.0	100.											
1	5	2	11.0	100.											
2	1	1	11.5	100.											
2	1	2	11.5	100.											
2	2	1	11.5	100.											
2	2	2	11.5	100.											
2	3	1	11.5	100.											
							SUM SHEARS (K)	TOP FIXED MOMENT FMT (K-FT)	BOTT FIXED MOMENT FMB (K-FT)	BOTT FIXED SHEAR FBQ (K)	SEE NOTE 2				

FORM G 7.4

154

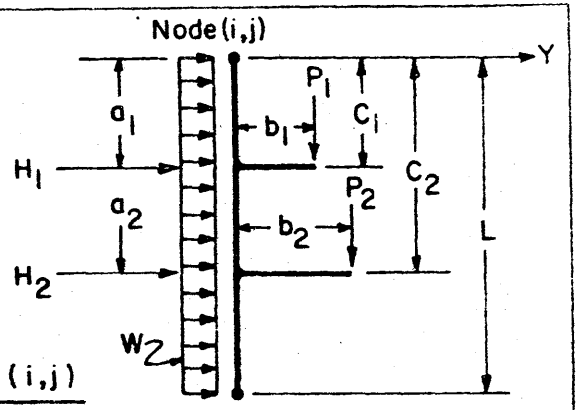
D

Table I (d). (Continued)

IBM DATA FORMAT SHEET — COLUMN DATA

JOB NO. B- _____ SHEET NO. 5 OF 5

DATE 5-6-64 BY NMD CH'K VLA



Notes

1. Forces Shown On The Column Are In The Positive Direction. Enter Minus Values For Opposite Direction.
2. Prismatic Member With Loads Shown On This Sheet-Enter Data Indicated Above Divider Lines. Non-Prismatic Members Or For Loads Not Shown—Enter R=0 And Data Indicated Below Divider Lines.

1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80																
2	3	2	1.1.5	1.00.												
2	4	1	1.1.5	1.00.												
2	4	2	1.1.5	1.00.												
2	5	1	1.1.5	1.00.												
2	5	2	1.1.5	1.00.												

I	J														
COL. NO.		LOAD COND. NO	COLUMN LENGTH L(FT)	RIGIDITY FACTOR R	UNIF. LOAD W (K/FT)	CONCEN. LOAD H_1 (K)	POSITION a_1 (FT)	CONCEN. LOAD H_2 (K)	POSITION a_2 (FT)	CONCEN. LOAD P_1 (K)	POSIT. b_1 (FT)	POSITION C_1 (FT)	CONCEN LOAD P_2 (K)	POSIT. b_2 (FT)	POSITION C_2 (FT)
							SUM SHEARS (K)	TOP FIXED MOMENT FMT (K-FT)	BOTT FIXED MOMENT FMB (K-FT)	BOTT FIXED SHEAR FBQ (K)	SEE NOTE 2				

FORM G 7.4

Scheme 1 - Analysis Results. -- The data sheets were keypunched and the problem analyzed on the computer. Table 2 (a) gives member end rotations/deflections for each load condition. Table 2 (b) gives member end moments/shears. The final moment/reaction diagram is given on Figure 5.

JOB NO. B-27500 CUSTOMER- PROJECT-
STEARNS-ROGER CORPORATION MULTISTORY RIGID FRAME PROGRAM - G7
SUBJECT- BIN SUPPT. FRAME C- SCHEME I BY NMD DATE 5-6-64

MEMBER ROTATIONS AND DEFLECTIONS

		Load Cond. 1 Vert. Loads	Load Cond. 2 Lateral Loads	
MEA	1	-.00034 Lt. Rot.	-.00002	
MEA	2	-.00069 Rt. Rot.	-.00006	11
MEA	3	0.00000 Defl.	0.00000	
MEA	4	-.00069	-.00006	
MEA	5	0.00000	-.00083	12
MEA	6	0.00000	0.00000	
MEA	7	0.00000	-.00083	
MEA	8	.00069	-.00006	13
MEA	9	0.00000	0.00000	
MEA	10	.00069	-.00006	
MEA	11	.00034	-.00002	14
MEA	12	0.00000	0.00000	
MEA	13	-.00002	-.00039	
MEA	14	.00007	-.00069	21
MEA	15	0.00000	0.00000	
MEA	16	.00007	-.00069	
MEA	17	0.00000	-.00083	22
MEA	18	0.00000	0.00000	
MEA	19	0.00000	-.00083	
MEA	20	-.00007	-.00069	23
MEA	21	0.00000	0.00000	
MEA	22	-.00007	-.00069	
MEA	23	.00002	-.00039	24
MEA	24	0.00000	0.00000	
MEA	25	-.00034 Top Rot.	-.00002	
MEA	26	-.00002 Bott. Rot.	-.00039	11
MEA	27	.03172 Defl.	.06168	
MEA	28	-.00069	-.00006	
MEA	29	.00007	-.00069	12
MEA	30	.03172	.06168	
MEA	31	0.00000	-.00083	
MEA	32	0.00000	-.00083	13
MEA	33	.00001	.10957	
MEA	34	.00069	-.00006	
MEA	35	-.00007	-.00069	14
MEA	36	-.03171	.06164	
MEA	37	.00034	-.00002	15

Beams

Columns

Table 2 (a). Member End Rotations/Deflections.

MEA	38	.00002	-.00039	15
MEA	39	-.03171	.06164	← defl. top row
MEA	40	-.00002	-.00039	
MEA	41	.00035	-.00157	21
MEA	42	-.03168	.16245	
MEA	43	.00007	-.00069	
MEA	44	.00030	-.00141	22
MEA	45	-.03168	.16245	
MEA	46	0.00000	-.00083	
MEA	47	0.00000	-.00083	23
MEA	48	.00001	.11457	
MEA	49	-.00007	-.00069	
MEA	50	-.00030	-.00141	24
MEA	51	.03175	.16249	
MEA	52	.00002	-.00039	
MEA	53	-.00035	-.00157	
MEA	54	.03175	.16249	25 ← defl. bott. row

Column

Lateral deflection - load cond. 2

.06164
.16249
δ = .22413"

Table 2 (a). (Continued)

JOB NO. B-27500 CUSTOMER- PROJECT-
STEARNS-ROGER CORPORATION MULTISTORY RIGID FRAME PROGRAM - G7
SUBJECT- BIN SUPPT. FRAME C- SCHEME I BY NMD DATE 5-6-64

FINAL MOMENTS AND SHEARS

		Load cond.1 Verti. Loads	Load cond.2 Lateral Loads	
EMS	1	-.21 Lt. Mom,	-70.59	
EMS	2	-1801.10 Rt. Mom.	-91.76	11
EMS	3	-471.67 Rt. Shear	-11.59	
EMS	4	1945.50	-47.43	
EMS	5	0.00	0.00	12
EMS	6	-392.22	-2.37	
EMS	7	0.00	0.00	
EMS	8	-1945.50	-47.37	13
EMS	9	-587.78	-2.36	
EMS	10	1801.10	-91.67	
EMS	11	.14	-70.54	14
EMS	12	-214.34	-11.58	
EMS	13	2.55	-127.40	
EMS	14	11.03	-153.78	21
EMS	15	.97	-20.08	
EMS	16	0.00	0.00	
EMS	17	0.00	0.00	22
EMS	18	0.00	0.00	
EMS	19	0.00	0.00	
EMS	20	0.00	0.00	23
EMS	21	0.00	0.00	
EMS	22	-11.07	-153.76	
EMS	23	-2.59	-127.38	24
EMS	24	-.97	-20.08	
EMS	25	.20 top. mom.	70.59	
EMS	26	24.23 bott. mom.	43.86	11
EMS	27	2.22 bott. shear	10.40	
EMS	28	-144.12	139.20	
EMS	29	42.55	-13.38	12
EMS	30	-9.23	11.43	
EMS	31	0.00	.03	
EMS	32	0.00	.01	13
EMS	33	0.00	0.00	
EMS	34	144.11	139.07	
EMS	35	-42.61	-13.52	14
EMS	36	9.22	11.41	
EMS	37	-.20	70.54	15

Beams

Columns

Table 2 (b). Member End Moments/Shears.

EMS	38	-24.23	43.81	15
EMS	39	-2.22	10.39	
EMS	40	-26.79	83.53	
EMS	41	0.00	-.01	21
EMS	42	-2.32	7.26	
EMS	43	-53.57	167.16	
EMS	44	0.00	-.04	22
EMS	45	-4.65	14.54	
EMS	46	0.00	.03	
EMS	47	.01	.05	23
EMS	48	0.00	0.00	
EMS	49	53.66	167.34	
EMS	50	-.02	.11	24
EMS	51	4.66	14.56	
EMS	52	26.83	83.58	
EMS	53	0.00	.02	25
EMS	54	2.33	7.26	

Columns

Table 2 (b). (Continued)

Scheme 11 - column pin-connected to top girder. -- All cards required to reflect continuity over the center column were keypunched and inserted in the original data deck, and the problem reanalyzed on the computer. Figure 6 shows the idealized frame and also the final moment/reaction diagram for the computer rerun.

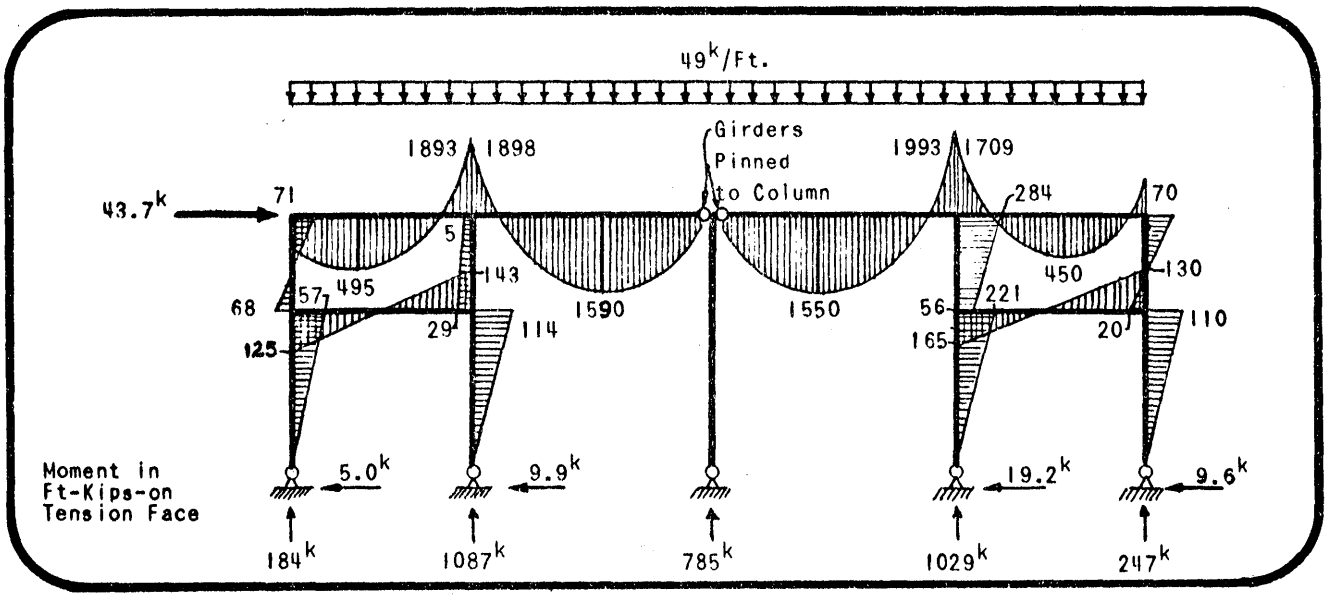


Figure 5. Moments/Reactions for Combined Vertical and Horizontal Loads - Scheme I.

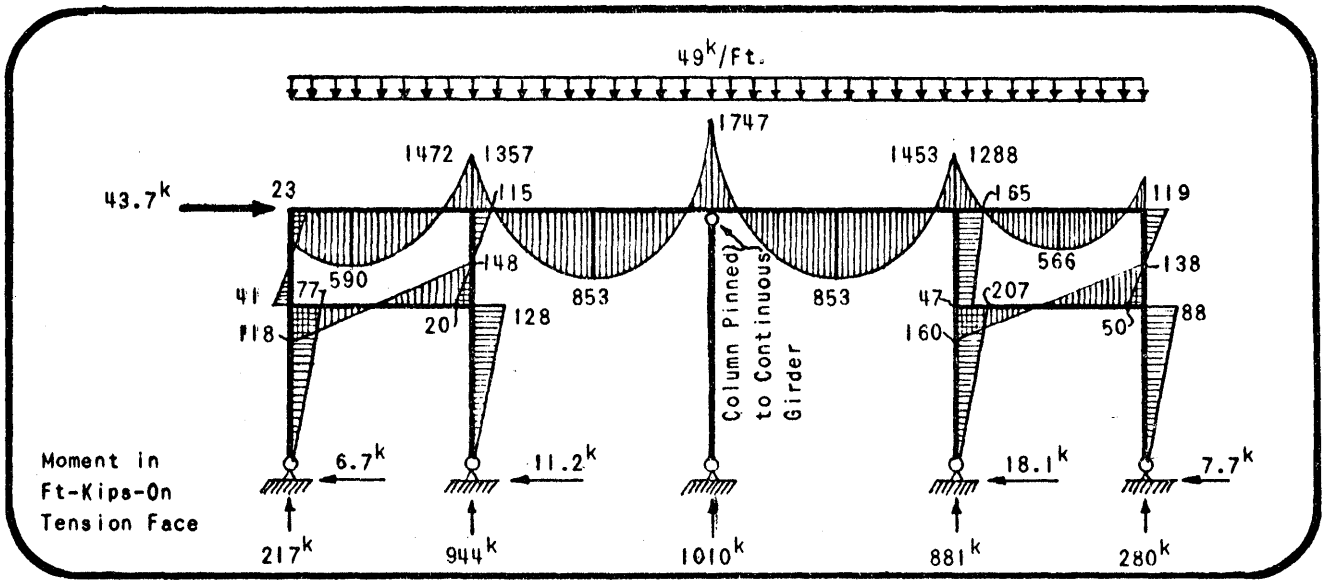


Figure 6. Moments/Reactions for Combined Vertical and Horizontal Loads - Scheme II.

Comparison of results. -- For combined vertical and lateral loading, it can be observed that Scheme II produced lower moments in the top girder (1747 ft-kip vs 1993 ft-kip for Scheme I). However, Scheme I produced a considerably lower center column reaction (785 kip vs 1010 kip for Scheme II).

2. Final Design

Final member sizes. -- Members were sized, based on the computer results for Scheme I, because of foundation size limitation. With the exception of the center column, base plates and foundations were designed for an assumed 50% fixity in order to reduce wind caused sway. The center column was designed as pinned at top and bottom, so that only axial load could be transmitted. Any column fixity would have produced overturning moments, necessitating larger center column foundations. All members were fabricated from plate. The steel was fabricated in Corpus Christi, Texas. Columns were stress relieved according to ASME Specifications. They had to be shipped to the nearest stress relieving furnace at Houston, Texas. There are approximately 250 tons of steel in each bin support (including floor framing), most of this is A441 steel.

No recognized standard procedures covering the design of the composite top girder could be found in the building codes, or available literature. This was due mainly to the massive size of the girders, and also to the heavy loading pattern. Much guidance and help was received from the American Institute of Steel Construction in designing shear connections for these girders.

Figure 7 shows the final frame with member sizes noted.

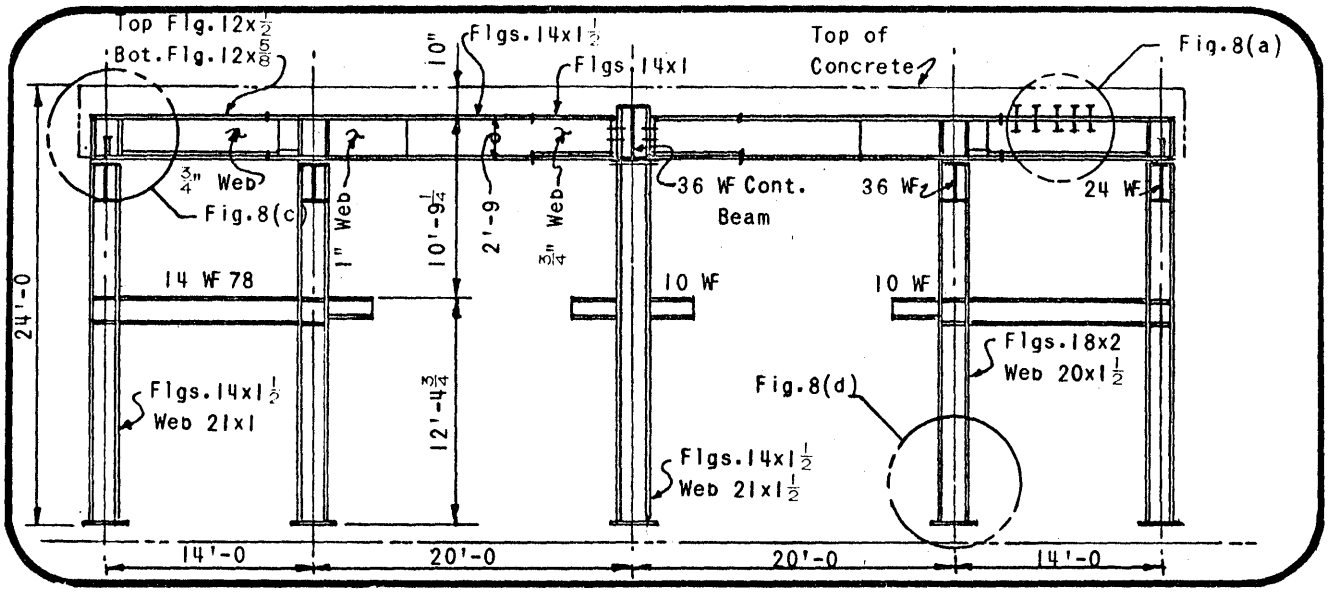


Figure 7. Elevation of Frame Showing Final Design Member Sizes.

Final computer run. -- Since Scheme I computer results were predicated upon assumed member stiffnesses and pinned column bases, the decision was made to rerun the frame using actual member stiffnesses. The purpose of this run was to give a final design check on the frame and to produce computer results compatible with the final design. All member sizes were found to be acceptable and no design changes were made.

Miscellaneous details. -- Figure 8 shows various details of the frame.

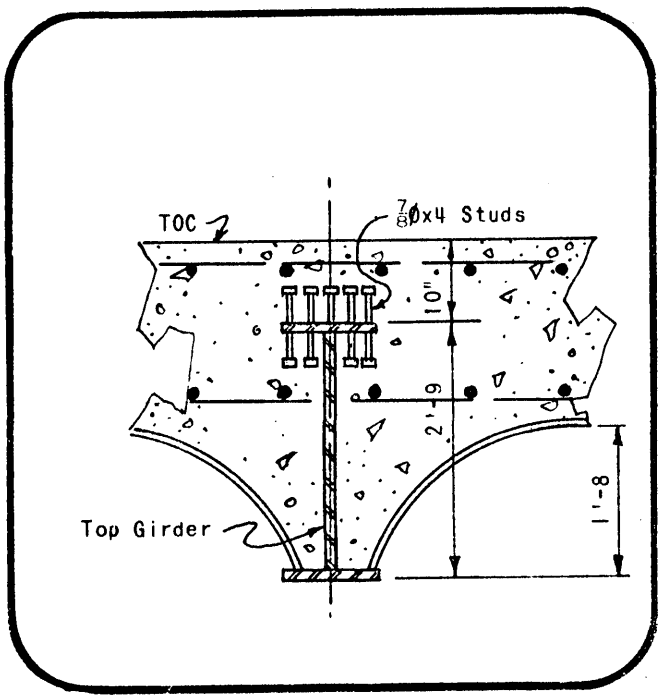


Figure 8 (a). Top Girder.

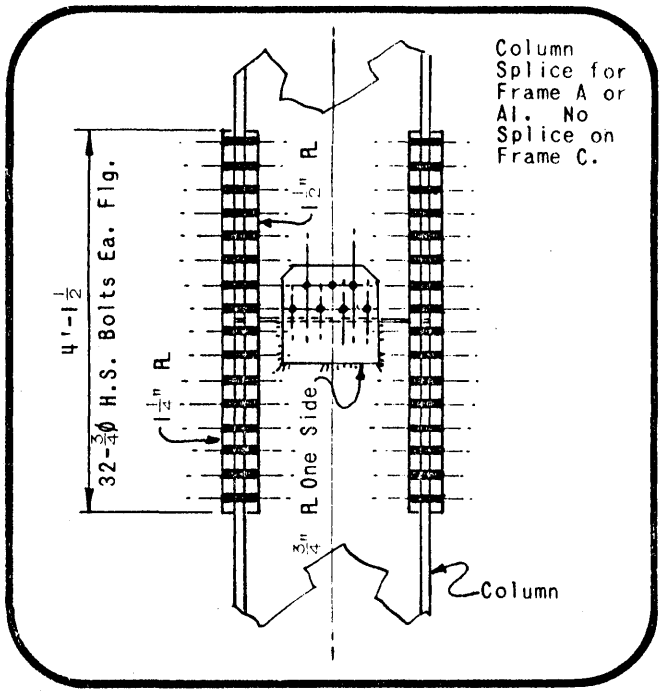


Figure 8 (b). Column Splice.

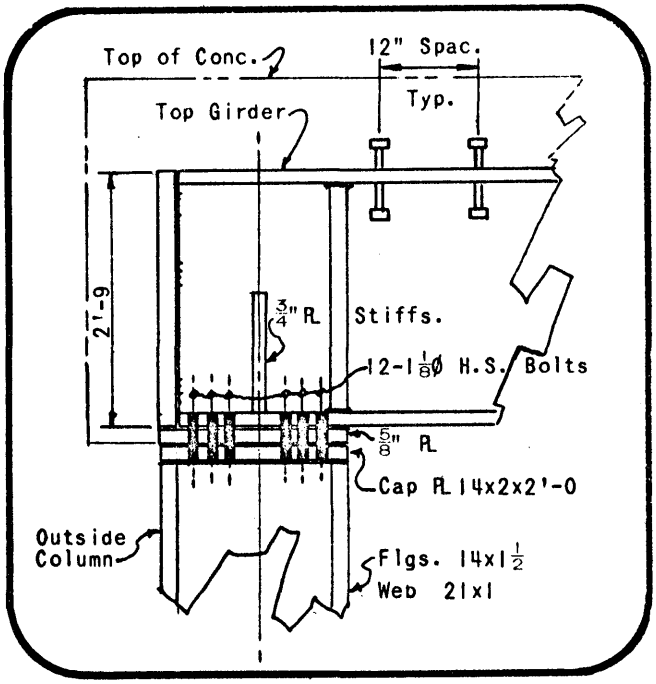


Figure 8 (c). Girder to Column Connection.

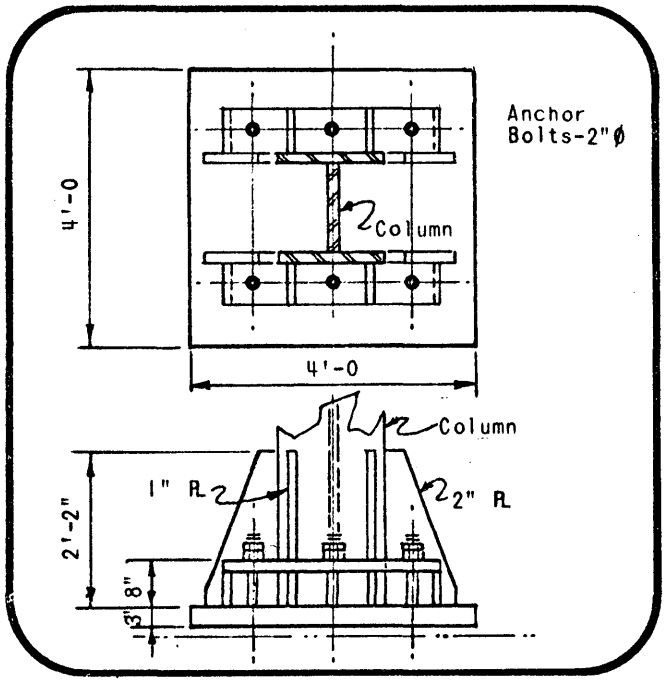


Figure 8 (d). Base Plate Detail.

LIBRARY, MAGAZINES, AND A COMPUTER

By Gale Ahlborn
Assistant Research Engineer
Institute of Transportation and Traffic Engineering
University of California, Berkeley

There has been a considerable amount of information published concerning libraries and automation, the term automation referring to anything from manual manipulation of punched cards to the use of computers with disk and tape files and remote communication equipment. We were interested in the use of computers in library operations but were unable to find out what has actually been accomplished or completed and whether these operations have been beneficial or not. There is little doubt that large libraries can make good use of computer equipment in their operations, but there has been some question about the use of computers with smaller libraries. A few examples of the use of computers with libraries are: Key-Word-In-Context indexes, catalog card production, producing book catalogs, circulation control, lists of recent acquisitions, and operations with library serials.

From the standpoint of making use of a computer, library activities can be generally considered to be the manipulation of many files of information -- a searching and sorting of various items of information for the purpose of performing some control on a library operation, producing an information list, or a combination of the two. These various library information files are of varying size and, most important, require updating at various intervals of time. The frequency of updating is a function of the particular file and the operation being performed with this file. As an example, if the operation is control of circulation, that is control and information concerning publications being checked in and out of the library, this file requires updating at least daily if not immediately as each operation occurs. On the other hand, a file concerning annual publications can be updated at a much less frequent interval, possibly only once per year.

Each computer installation differs from another in equipment, size, personnel, and operation. The same applies to libraries. Each library is different in size, equipment, personnel, and operation.

Our particular computer equipment is a 1620 Model I system, twenty thousand positions of core storage, one disk drive with punch card input and output. Supporting off-line equipment for this system are key punches, sorter, and a 407 accounting machine for printing. Our partner in this operation is the Transportation Engineering Library, a special purpose library. Both computer and library are located at the Richmond Field Station about five miles from the Berkeley campus of the University of California. The primary use for this computer is educational. It is used by scheduled engineering classes

and graduate students in their advanced degree research. The development of the use of this equipment in the library's operation has had a low priority and therefore has progressed rather slowly.

Our first venture into this kind of library automation has been to work with the library's serial file. Since I am a library user rather than a librarian, I am substituting the word "magazine" in place of "serial," although I have been informed that this is not entirely correct. There are various items of interest concerning magazines, such as, the frequency of issue, the general subject matter of each magazine, the cost of subscription, the length of subscription, to whom the magazine is circulated, and in what order. At the end of each year some magazines are discarded, others are held a certain number of years, while others are retained indefinitely. For those magazines that are retained, there are other items of interest, such as, the number of issues to be bound together, who does the binding and what type of binding is used. The magazine file of the Transportation Engineering Library consists of about five hundred and fifty magazines, and there are about thirty possible items concerning each magazine that are of interest to the library.

When we first started, we considered developing a rather complete system of computer operation with the magazine file. This system would have the magazine file on the disk storage at all times, available for on-line updating and producing various reports to provide information and control of the library's magazine operation. Although we felt this system had merit, we were concerned as to whether it would be justified for a small computer-small library operation and also concerned about the lack of experience on both sides in this area of library automation. We finally decided that it would be better to develop a manual system of operation first, use this system for a period of time, review our experience, then determine whether to continue with the same or a different manual system or to switch over to a complete computer operation.

The manual system consisted of having the magazine file punched into a deck of cards, then using the mechanical sorter and accounting machine to produce the desired reports. With this system, the computer was to be used only as necessary to supplement the operation. Three cards were used for the information concerning each magazine. The first two cards contained information that was of interest to both the library and library users. The information on the third card was only of interest to the library. Numeric codes were duplicated into a common area on all three cards for the purpose of sorting. The information punched into the cards outside of the sorting area was either in plain language or a form of alphabetical coding so that the information is relatively easy to understand when printed directly on the accounting machine.

Five different reports are produced using this manual system. The largest report is the "Users Report" that contains nine different listing from the magazine file. These lists include an alphabetical list, newsletters, statistical sources, foreign periodicals arranged by country of origin and language, subject index, indexing and abstracting services. In all, the report is sixty-five pages in length. Since there are about 250 copies of this report distributed, offset masters for the press are printed directly from the cards on the accounting machine. In addition to the five reports presently produced, there are at least five more that are desired which we are hoping to avoid producing until we revise our present magazine file system.

We have used this manual system for approximately a year with no major changes being made to the system. Several items of information about the magazines were deleted from the file, while other items were added. It became necessary to make use of the computer almost from the very beginning of this operation for several reasons. First, the computer is very efficient at checking the validity of codes punched in the cards. We use both the computer and a manual checking of the list to detect errors. Second, we found that it was desirable to use various items of information in various combinations which would involve a fairly complex mechanical sorting. The computer was used quite easily for doing this type of operation.

We found that updating of the magazine file consisted mostly of making many small changes throughout the file rather than having to make large changes. The frequency of updating with this file is dependent on the frequency of producing reports. Considering all the various types of reports that our library would desire from this file, it would not be necessary to produce reports any more frequently than once per month.

Our conclusion, after working with this manual system, is that the magazine file should be changed over to a computer system. The system that we have devised and is being developed at the present time is as follows: The magazine file will be retained on cards in the library where they can be easily referred to and updated as needed. When reports are required, the card file will be loaded onto the disk working area and then the reports generated. While this system sounds very similar to what we had originally considered, there actually are large differences between the two systems. We know now that if we had developed the original system, that there would have been considerable difficulty with it, some portions being practically unworkable. We fee now that we are developing a very practical and workable system that will have sufficient flexibility. This system will set the pattern for the additional work we expect to do with the library.

ACCURATE SOLUTION OF

SYSTEMS OF LINEAR EQUATIONS

Presented by

David R. Musser
Hoblitzelle Computer Center
Austin College
Sherman, Texas

WESTERN REGION SUMMER MEETING OF COMMON

Denver Hilton Hotel, Denver, Colorado

July 8, 1966

Accurate Solution of Systems of Linear Equations

This paper is concerned with the solution of the matrix equations

$$Ax = b, \tag{1}$$

$$AD = I, \tag{2}$$

where A is a general n x n matrix, assumed to be non-singular.

Eqn. (1) is the matrix formulation of the system of n equations in n unknowns

$$\sum_{j=1}^{n} a_{ij}x_j = b_i, \quad i = 1,2,\ldots,n.$$

Eqn. (2) is the problem of finding the inverse $D = A^{-1}$ of A.

Although the solution of (1) is given by $x = A^{-1}b$, methods of computing x exist which involve less calculation than the calculation of A^{-1} and multiplication by b. On the other hand, the solution of Eqn. (2) itself may be treated as an extension of Eqn. (1). Again, however, there are methods of computing A^{-1} which require less calculation than this extension. Thus optimum efficiency requires separate methods of dealing with these two problems, although there are many operations which are basic to both. (Considerations of efficiency will, in this paper, be almost entirely from the standpoint of the number of calculations--and thus the time--required to obtain the solution to a given degree of accuracy, rather than in terms of storage requirements.)

Even if a method is chosen which is algebraically best suited to the given problem (solution of systems of equations or the inverse problem), it may not give the accuracy desired in a particular application, because of the nature of the matrix A--i.e. A may be almost singular, or "ill-conditioned." Worse still, the results calculated may offer no indication of the condition of A, although in physical applications it is often important to know how close to singular A is, so that the effect of imprecision in the elements of A can be estimated. If A is ill-conditioned then even slight imprecision in the elements of A may mean gross imprecision in the solution.

A standard method of obtaining high accuracy in computer solutions of matrix equations is to carry out all calculations in double precision. However, this requires about four times as much calculation as single precision and twice as much storage. Also no knowledge of the condition of A is obtained, and the amount of calculation is the same whether A is well- or ill-conditioned.

A better procedure than blanket use of double precision appears to be to calculate by a direct method an initial approximation to the solution in single precision, and then improve this solution by a method of successive refinement. The latter method does require some double precision, but it turns out that only double precision additions are required, and none of the elements of the matrix A or the solution need be stored in double precision.

With this method the number of iterations and thus the amount of calculation required depends on the condition of A and will be less for a well-conditioned matrix. In addition, the number of iterations does give a valuable indication of the condition of A.

A minor modification to this procedure is to do some of the calculation in the direct method in double precision so that a better initial approximation is obtained. In this case, for a given matrix fewer iterations will be required. Again, accuracy can be significantly improved using only double precision additions, if a proper ordering of the calculations is made.

A direct method which accomplishes this ordering is the Crout method, one of several "compact" schemes, so-called because they require no intermediate modification and restorage of the elements of A.

This paper will not go into the mathematical details of either the Crout method or the iterative improvement method; rather, its remainder will be devoted to a general description of the Fortran II subroutines which have been written to implement this method on the IBM 1620 computer. Special attention will be given to the subprogram which is the heart of the method: the DOT function subprogram which is coded in SPS to do the necessary double precision arithmetic. Using these routines solutions can usually be obtained which are correctly rounded to single precision (eight digits), even when A is very ill-conditioned.

The mathematical details of the methods used are contained in a thesis which the author wrote at Austin College. The essential parts of this thesis will be included in the program writeup when the set of routines is submitted to 1620 program library. The ideas behind these routines and the thesis were, of course, not original. The literature on the subject is extensive. Primary stimulus for the project came from a report by Cleve Moler of the Jet Propulsion Laboratory in California, entitled "Numerical Matrix Inversion" (JPL Technical Report No. 32-394, March 15, 1963).

General description of the Fortran II programs

The structure of the programs was designed to make them flexible and convenient to use without making it necessary to waste time or storage by providing a more general program than is necessary for a particular application.

Thus the implementation has a hiearchial structure, being composed of nine subprograms, some of which are used primarily to organize the others for the special task to be performed. The basic subprograms and their uses are as follows:

DOT: accumulate inner products in double precision and round the result to single precision (used by all of the basic routines below except INOUT).

FACTOR: factor matrix A into product LU using Crout method with partial pivoting.

SOLSYS: solve $Ly = b$ and $Ux = y$.

SIMPRV: improve the accuracy of the solution vector x (uses SOLSYS).

INVA: form inverse D of A from L and U.

IMPRUV: improve the accuracy of the inverse D.

INOUT: input and/or output a matrix.

These basic routines may be used directly by the programmer in his own Fortran II program, or more conveniently by use of one the following subprograms:

SOLVE: solve $Ax = b$ for x (uses FACTOR, SOLSYS, and SIMPRV).

INVERT: find inverse D of A (uses FACTOR, INVA, and IMPRUV).

Finally, a main program is provided which can input a matrix, perform either the inversion or solution of equations, or both, and output the results. The basic subprograms are used directly instead of SOLVE and INVERT however, to avoid repetition of the factorization of A.

DOT function subprogram

The methods used in the Fortran routines were chosen so that all of the critical arithmetic would be concentrated in a single type of operation: the computation of inner or "dot" products of vectors:

$$a \cdot b = a_1b_1 + a_2b_2 + \ldots + a_nb_n = \sum_{i=1}^{n} a_ib_i.$$

In practice there are two ways of computing such sums of products.

(1) Round (or truncate) to single precision after each multiplication and each addition.

(2) Retain the sum of products in a double length accumulator until the entire dot product has been formed, then round to single precision.

It should be obvious that the amount of round-off error incurred would be much less in the latter method than the first. Yet only double precision additions are required, since the multiplications are of single precision operands (giving a double length product as with any single precision multiplication).

The special function subprogram called DOT was written for the 1620 to compute inner products using the latter method of computation.

The structure of DOT was patterned after that of a routine written for the IBM 7090, described by Moler. It is structured so that it is possible that one or both of the vectors of which the dot product is to be formed may be a row or column of a matrix. (See Figure 1 for an example of the use of DOT to perform a matrix multiplication.) Provision is made for adding a single number to the accumulated product (actually the accumulator is set equal to this number before the product accumulation begins). If the number of terms in the summation is specified to be zero, then this single number becomes the value of the summation. These features make the DOT suprogram a valuable programming aid, since a single use of it can replace several statements (as in Figure 1), and the special case where the number of terms in the summation is zero is automatically taken care of.

All of these structural features can be implemented by a simple subprogram in the Fortran II language. In most Fortran II systems, however, double precision arithmetic is not allowed. This was the case with the 1620 Fortran II system, even though the 1620 is a

variable word length and the Fortran II system does to an extent take advantage of this feature. In 1620 Fortran II programs it is possible to vary the length of the floating point word mantissas from 2 to 28 digits, but unfortunately this variation is permitted only from program to program and not with a single program or set of subprograms. That is, if 8 digit mantissas are desired for most numbers stored by the the program, there is no way to use any numbers with longer mantissas. Thus a special machine language routine had to be written to implement DOT on the 1620.

Conclusion

The DOT subprogram and other subprograms have been extensively tested at the Austin College Computing Center. These tests bear out what is mathematically evident: that the use of the DOT method of accumulating inner products contributes significantly to the accuracy obtainable from the Crout method, and is essential to the proper convergence of the iterative improvement scheme.

Thus the DOT function would seem to be a very useful tool to have available for matrix work, where solutions of known high accuracy are desired. In fact the DOT function potentially has many other uses, in any calculations in which the critical arithmetic can be concentrated in the calculation of inner products. As Moler points out, for example, the complete eigenvalue problem is thus formulated by Householder's method.

* * *

Algebraic notation: $$C_{ij} = \sum_{k=1}^{N} A_{ik} B_{kj}$$

Fortran coding:

```
      SUM = 0.
      DO 10 K = 1,N
   10 SUM = SUM+A(I,K)*B(K,J)
      C(I,J) = SUM
```

Fortran with DOT function:

```
C(I,J) = DOT(N,A,I + MA,MA,B,1 + J*MA,1,0.)
```

* Figure 1 *