1103 CENTRAL EXCHANGE

NEWSLETTER NUMBER 9

June 1956

PX 71900-9

Remington Rand Univac
DIVISION OF SPERRY RAND CORPORATION
1902 WEST MINNEHAHA AVE. ST. PAUL W4, MINNESOTA

Newsletter Number 9

June 1956

EDITOR'S PAGE

Coding and checkout of Trans-Use is progressing quite satisfactorily. Trans-Use is a routine which will translate programs written in USE language into 1103A machine language in a format ready for execution or assembly. The USE-Compiler will be a much more powerful instrument capable of doing a variety of functions. For most problems, especially fixed point programs, Trans-Use will be a useful instrument even when the Compiler is available. Trans-Use is being prepared by Holloman Air Development Center.

A minimum Service Routine Library for the 1103A is about 75% complete here at St. Paul. This minimum library will include paper tape input-output routines and several diagnostic routines for program debugging purposes.

On page 1 of the SNAP Sampler (RW-140) write-up, the last sentence of paragraph b should be replaced by: "Restoring the library from magnetic tape loads an all zero word into cell 71777_b. If this word is not changed a complete trace of all SNAP commands is automatically performed."

Future contributions and communications to the Central Exchange should be addressed to Leo Kennedy, Systems Analysis Department, who has assumed the duties of Central Exchange editor.

Leo Kennedy
Systems Analysis Department

NEWSLETTER NUMBER 9

JUNE 1956

REPORTS

CONVAIR — An 1103 program for computing Eigenvalues and Eigenvectors of Real, Symetric Matrices has been completed. This program is designed to determine all vectors Y_i and all scalars λ_i which together with a given real, symetric matrix A of order $N \leq 64$ satisfy the relationship $AY_i = \lambda_i Y_i$. The mathematical technique employed is based on the Hestenes - Karush gradient methods. The program is made to take advantage of a matrix with large numbers of zero elements by representing it in dehydrated form, i.e., with blocks of zero elements replaced by flags indicating the number of zeros removed.

RAMO-WOOLDRIDGE — A table of contents for the Ramo-Wooldridge Utility Library for the 1103 is enclosed in this Newsletter. In connection with this, the following communication was recieved from Ramo-Wooldridge: "We now have our library available on a self loading deck of binary cards. We would be glad to supply a copy of this deck to anyone, together with complete instructions for loading the deck in the 1103 and a list of locations occupied by each routine. With this information the various output routines which are part of the library could be used to obtain octal or binary cards or bioctal paper tape for any or all routines."

LOCKHEED — As a first step in the direction of exchanging information about the organization of individual computing centers, W. W. Leutert, Head of Mathematical and Computer Service Department (Dept. 66-10) Lockheed Missile Systems Division has submitted an organizational description of his department for Central Exchange distribution. It is hoped that this first step will stimulate the flow of such worthwhile information among the various computer installations.

WRIGHT FIELD — A decimal output routine for the ERA Line Printer has been completed. Decimal digits must be presented to the routine in coded form and the speed of the output is limited only by the Line printer itself (150 lines per minute, 92 characters per line).

NEWSLETTER NUMBER 9
JUNE 1956

REMINGTON RAND UNIVAC Enclosed in this Newsletter is a report on "A Linear Programming Routine for the 1103 Computer" which is being developed at St. Paul.

A preliminary report on "A Multiple Correlation and Regression Program for the 1103" is enclosed. This program has been used successfully at St. Paul on several customer production problems.

Also included is a description of the Utility Routine Library for the Serial 9 1103 at St. Paul. This library (RR-126) consists of three main parts: (1) Service Routine Library; (2) Regional Coding Routine; (3) Library Routine.

A few inquiries have been directed to us concerning the action of the LEFT TRANSMIT instruction (LTjkv) of the 1103A (1) when $7 \geq j \geq 2$, and (2) when $k > 177$. The quantity "j" in this instruction at present consists effectively of the one bit, u_{12}, instead of the usual three bits, u_{14}, u_{13}, u_{12}. Hence for $j = 2$, 4, or 6, effectively $j = 0$; and for $j = 3$, 5, or 7 effectively $j = 1$. No anomalies arise when $k > 177$ as is the case for the Split Instructions and Left Shift Instructions in A and Q.

We would like to take this opportunity to review some of the actions of the 1103A "interrupt" signal during a repeated operation. Since the Repeat Sequence by-passes Main Pulse 6 and 7, the "interrupt" will not take effect until the Repeat Sequence is terminated. For Normal Termination, the "interrupt" will take effect on MP6 of the execution of the jump instruction stored at F_1. When a jump condition is met during a repeated Threshold or Equality Jump, the "interrupt" will take effect on MP6 of the Jump Termination Sequence. Hence, for both the Normal and Jump termination of the Repeat Sequence, the "interrupt" does not become effective until PAK contains the address of the next instruction in the otherwise un-interrupted program.

PX 71900-9-(iii)

ENCLOSURES

OR-124	Normal Derivate Routine
RW-125	Linear Matrix Equation Solver
RR-126	Utility Routine Library, Regional Coding Routine, Library Routine
RR-127	Multiple Regression and Correlation Routine
OR-128	Magnetic Drum to Magnetic Tape Dump
CV-129	Card Read and/or Punch Routine
CV-130	Card Punch Routine
CV-131	Two Cycle Read Only Card Routine
CV-132	Solution of Simultaneous Linear Equations by the Method of Crout
CV-133	Square Root - Floating Point
CV-134	Cube Root - Floating Point
RW-135	Fixed Point Card Output Subroutine
RW-136	Stated Point Card Output
RW-137	Octal Card Dump
WF-138	CHIP - a Floating Point Interpretive Subroutine
WF-139	Polynomial Expansion, $\sum_{n=0}^{i} a_n x^n$
RW-140	SNAP Sampler Trace
RW-141	SNIP - Interpretive Floating Point Package - Complex
RW-142	Eigenvector, Eigenvalue Routine for Real Symmetric Matrices
RW-143	Floating Point Gill Method
RW-144	Floating Point Sine - Cosine
RW-145	Standard Atmosphere Calculation
RW-146	Manual Inspection and Insertion

REVISIONS

CV-39	Floating Point Card Output

22 March 1956

OPERATIONS RESEARCH OFFICE
7100 Connecticut Avenue
Chevy Chase, Maryland

Title:		Normal Deviate Routine
Format:		Standard Form
Storage:	a) Total:	01000 - 01030, 31 octal
	b) Instructions:	01000 - 01016, 17 octal
	c) Constants:	01017 - 01024, 6
	d) Temporary Storage:	01025 - 01030, 4
Alarm Conditions:		
	Alarm when	(01030) $\leq$ 0. This location must be supplied a randomly selected positive number before the routine is entered the first time.
Timing:		12 milliseconds per deviate
Exit Condition:		(A) = deviate scaled 2^{32}
Range:		deviate will be in range $\pm$ 6
Coded and Machine Checked By:		F. R. Urbanus

PX 71900-9-(124)

Description:

The routine provides a means for drawing "at random" from an approximately normal distribution having a zero mean and a variance of 1.

The routine is based on a consequence of the Central Limit Theorem of Mathematical Statistics, which states that a distribution of sums of uniformly distributed random variables approaches the normal distribution as the number of variables summed is increased. In this routine twelve random variables, each from the same uniform distribution and in the range 0 to $2^{24}-1$, are summed and then normalized (i.e., reduced by the theoretical mean of the distribution and divided by the square root of the theoretical variance of the distribution). The resulting normalized deviates are thus distributed approximately normally with mean 0; variance 1.

The accuracy of the approximation has been measured by collecting 5 samples of 1000 deviates each and checking the distributions by statistically testing the hypothesis that the distributions formed by the deviates are not significantly different from the normal distribution. For the results of these tests, see the section titled "Accuracy of Approximation".

If $S_i = P_{i,1} \pmod{2^{24}-1} + P_{i,2} \pmod{2^{24}-1} + \ldots + P_{i,12} \pmod{2^{24}-1}$,

where the P's are "pseudo-random" numbers in the range 1 to $2^{35}-32$, *

then the mean of S is

$$\bar{S} = \frac{12(0+2^{24}-1)}{2} = 6\,(2^{24}-1)$$ and the variance of the sum is

$$\sigma^2(\text{sum}) = 12\ \ \sigma^2(\text{uniform distribution})$$

$$\sigma^2(\text{sum}) = 12\left[\frac{\sum_{i=0}^{2^{24}-1} i^2 \;-\; 2^{24}\,\bar{S}}{2^{24}}\right]$$

$$\sigma^2(\text{sum}) = (2^{24}-1)\,(2^{24}+1)$$

The normalized sum, or deviate, is then

$$\frac{S-\bar{S}}{\sigma} = \frac{S-6\,(2^{24}-1)}{\sqrt{(2^{24}-1)(2^{24}+1)}} \sim \frac{S}{2^{24}} - 6 = S'$$

Since $0 \leq S \leq 12\,(2^{24}-1)$

$$-6 \leq S' \leq 6$$

Each deviate, then, can be no larger in absolute value than 6. Almost all (99.7%), however, can be expected to fall within the range ± 3.00

* The secondary modulus, $2^{24}-1$, was chosen arbitrarily to provide a wide range for the sums, and a fine gradation for the deviates. Furthermore, it was desired to have the secondary modulus equal to 2^n-1, in order to reduce the number of divisions required in the routine.

Accuracy of Approximation:

The following frequency distributions were compiled by the 1103. Each contains 1000 deviates. The mean, variance, measure of skewness, and measure of kurtosis for each distribution were tested statistically and found not to differ from the measures of the normal distribution (0, 1) by a significant amount.

Deviates	(1)	(2)	(3)	(4)	(5)	Total
-3.86 to -3.47	0	0	0	1	0	1
-3.47 to -3.09	1	0	2	2	0	5
-3.09 to -2.70	2	6	4	0	4	16
-2.70 to -2.31	5	10	13	13	3	44
-2.31 to -1.93	18	25	12	9	15	79
-1.93 to -1.54	44	31	30	36	37	178
-1.54 to -1.16	61	65	61	67	70	324
-1.16 to -0.77	103	104	83	99	101	490
-0.77 to -0.39	130	112	139	144	133	658
-0.39 to 0	155	141	135	168	145	739
0 to 0.39	151	147	164	152	148	762
0.39 to 0.77	137	147	128	115	109	636
0.77 to 1.16	82	90	101	85	101	459
1.16 to 1.54	56	64	66	56	74	316
1.54 to 1.93	29	28	38	36	36	167
1.93 to 2.31	11	21	15	15	15	77
2.31 to 2.70	9	6	7	7	5	34
2.70 to 3.09	6	3	1	0	4	14
3.09 to 3.47	0	0	1	0	0	1
3.47 to 3.86	0	0	0	0	0	0
Mean	-.046	-.031	+.012	-.060	-.008	-.024
σ^2	1.00	1.10	1.03	0.98	1.02	1.02
γ_1 (skewness)	0.088	0.090	0.138	0.023	0.052	0.076
γ_2 (kurtosis)	3.07	3.00	3.18	3.08	2.78	2.90

Because it appeared that the mean of the distribution might have some negative bias, a further test was made in which 200 samples, each representing the mean of 500 deviates, were compiled by the 1103. The mean of these means was +.0023, indicating that there is evidently no reason to suspect a biased mean.

Storage Address	Order	Function of Order
01000	37 76000 76002	Alarm exit
01001	45 00000 30000	Normal exit
01002	11 01017 01025	Set up index
01003	13 01020 01026	$(-\bar{S}) \longrightarrow (01026)$
01004	11 01030 20000	Random Number $\longrightarrow$ (A)
01005	42 01024 01000	Alarm , (01030) ≤ 0
01006	71 01030 01021	$RN \cdot 5^{13} \longrightarrow (A)$
01007	73 01022 10000	$R.N.5^{13} (\text{mod } 2^{35}-31) \longrightarrow (A)$
01010	11 20000 01030	Store new R.N.
01011	11 01023 10000	Mask $\longrightarrow$ (Q)
01012	51 01030 01027	$R.N. (\text{mod } 2^{24}-1) \longrightarrow (01027)$
01013	21 01026 01027	$(S-\bar{S}) \longrightarrow (01026)$
01014	41 01025 01006	Through 12 times?
01015	54 01026 00010	$(S-\bar{S}) * 2^{8} = S' \cdot 2^{32} \longrightarrow (A)$
01016	45 00000 01001	To Normal Exit
01017	00 00000 00013	Index

01020	00 05777 77772	$\overline{S}$
01021	01 10604 71625	5^{13}
01022	37 77777 77741	$2^{35}-31$
01023	00 00777 77777	$2^{24}-1$
01024	00 00000 00001	1
01025	[00 00000 00000]	Store index
01026	[00 00000 00000]	Store Sum
01027	[00 00000 00000]	Temporary Storage
01030	[00 00000 00000]	Current R.N.

THE RAMO-WOOLDRIDGE CORPORATION
Los Angeles 45, California

Linear Matrix Equation Solver (AX = B)

Specifications

Identification Tag:	MTI-0
Type:	Subroutine available on cards for assembly
Storage:	217 instructions, addresses 10M00 (00M00) thru 10M51 (0M51) 11M00 (01M00) thru 11M37 (01M37) 12M00 (02M00) thru 12M63 (02M63) 13M00 (03M00) thru 13M62 (03M62) 12 constants in program, addresses C1N00 (C0N00) thru C1N11 (C0N11) Temporary storage used, but not stored in program (See Text). 229 words total program storage. The constant pool and temporary storage pool are used by this routine.
Program Entrance:	Address 10M02
Program Exit:	Address 10M01
Alarm Exit:	The alarm exit is used by this routine.
Machine Time:	For all storage in ES time is approximately (in milliseconds): $.3\,n^3 + .9n^2m + 1.7n^2 + .3m^2 + 2.5nm + 1.8n + 1.6m + 2.7$ For temporary storage (see text) on drum add approximately (in milliseconds): $.04\left[n^3 + 4n^2m + 3n^2 + 10mn\right] + 51$
Mode of Operation:	Fixed point

Coded by:	W. L. Frank	October 25, 1955
Code Checked by:	W. L. Frank	November 15, 1955
Machine Checked by:	W. L. Frank	November 17, 1955
Approved by:	W. F. Bauer	November 30, 1955

Description

This subroutine solves the linear matrix equation AX=B, where A is a non-singular matrix of size nxn and B has the dimensions nxm. The solution, $X=A^{-1}B$, is a matrix of size nxm. For the special case, when B is the identity matrix (I), one obtains the inverse of the matrix A. Otherwise, one can solve m sets of n simultaneous linear equations in n unknowns.

Considerable flexibility is afforded the programmer with respect to the storage of the matrices A, B and the answer X. The programmer must code two auxiliary routines as follows:

(a) The first must provide successive rows of the augmented matrix $[A, B]$. (When B=I, one only need supply rows of A). Each row, consisting of (n+m) elements (or n elements when B=I), must be set up in the fixed location immediately following the subroutine. This data must be scaled at 2^{35} and be such, that for all elements a_{ij} of $[A,B]$

$$\left|a_{ij} \cdot 2^{35}\right| \leq 2^{34}$$

In the general case, for B≠I, the rows of $[A,B]$ may be scaled independently. However, in the case of inverting a matrix, it is necessary that the entire matrix be scaled by the same factor.

(b) The second auxiliary must take the successive columns of X, found in the n cells immediately following the routine, and either store them internally or punch them out. Since the columns of X are independently calculated, each has an associated scale factor (scaled at 2^0). This parameter positions the binary point, (assuming the input matrices are scaled at 2^{35}) and is to be found in the (n+1)st cell following the routine. If one has inverted a matrix, and if the input rows were originally scaled by 10^p (or 2^p), then the output columns must be re-scaled by 10^p (or 2^p).

These auxiliary routines are automatically entered n and m times respectively by RJ instructions. The subroutine sets up these two RJ instructions from information gleaned from the parameters of the entry. This procedure allows storage of A,B and X on ES, MD, magnetic tape or externally on cards or tape. It is also possible to generate the elements of successive rows when a functional relation exists.

In addition to the 229 words of storage needed by the subroutine, it is necessary to provide 2(n+m) cells temporary storage immediately following the subroutine, and a block of $\frac{n(n+1)}{2} + nm$ cells, either all on ES or all on MD.

Operating Instructions

1. Entrance to the subroutine is made by the following orders (B≠I):

p	RJ	00M01 00M02
p+1	00	00X00 00Y01
p+2	--	uuuuu vvvvv
p+3	--	----- xxxxx

where 00M00 is the location of the first word of the subroutine

00X00 is the location of the first word of the first auxiliary

00Y01 is the location of the second word of the second auxiliary

uuuuu = m (number of columns of B)

vvvvv = n (number of rows of A)

xxxxx = is the location of the first cell of the block of $\frac{n(n+1)}{2} + nm$ cells all in ES or all in MD.

2. For the case when B=I, the p + 1 word must be 40 00X00 00Y01

PX 71900-9-(125)

3. The auxiliary routines must be available and coded so that they can be entered with

   RJ 00X00 00X01

   and RJ 00Y00 00Y01 respectively.

   This implies that the first and second words of both auxiliaries are exit and entrances respectively.

Alarm Conditions

Two alarm conditions can result:

1. A test is made to see that all elements, a_{ij} of the input rows are within the limits

   $$\left| a_{ij} \cdot 2^{35} \right| \leq 2^{34}$$

   If this is violated the alarm routine ALR-1 is entered and "alarm -xxxxx" is printed where xxxxx-3 is the address of the cell from which the subroutine was entered.

2. If a singular matrix is detected in the process of inversion, the alarm routine ALR-1 is entered and "singul-wwwww" is printed where wwwww-3 is the address of the cell from which the subroutine was entered. The routine can not, however, detect all singularities due to round-off errors (see below).

   Starting again at xxxxx+1 will cause the rest of the main program to be obeyed.

Machine Time

The machine time is as indicated on the first page when all operations are carried on in ES. This time is exclusive of the times taken by the auxiliaries. In case the block of $\frac{n(n+1)}{2} + nm$ words are stored on MD, the time must be increased by the terms indicated.

These times are approximate and will be a minimum in most cases.

Sample computation times for matrices of order 27 and 99 were respectively 53 seconds and 30 minutes.

Mathematical Method (Gauss elimination method)

Elementary row operations are performed on the matrix A reducing it to an upper triangular matrix $\overline{A}$. At the same time, these operations are performed on the matrix B giving a new matrix $\overline{B}$. A partial floating point arithmetic is maintained, in that the rows of the augmented matrix $[A,B]$ are always kept within the limits such that the largest element of the row (in absolute value) lies in the interval

$$2^{34} > |a_{ij} \cdot 2^{35}| \geq 2^{33}$$

In addition, before eliminating, leading elements of two rows are compared and the element of largest magnitude becomes the pivotal point.

Next, successive columns of $\overline{B}$ are taken and the equation $\overline{A}X=\overline{B}$ is solved by the back substitution procedure.

Singularities in A are detected if a zero appears on the diagonal of $\overline{A}$. Since round-off errors can prevent this from occurring, one must inspect the size of the scale factor if A is suspected of being singular. Ill conditioned matrices will cause the scale factors to be very small. That is, the elements of X will be very large.

Accuracy

The accuracy in the result is a function of the condition of the matrix A. Seven to eight decimal place accuracy was obtained for matrices of order 10 to 16. A matrix of order 39 and 99 yielded 7 and 6 place accuracy respectively.

```
D           10M00  00100                                  144  00  00000  00000
D           11M00  00152                                  230  00  00000  00000
D           12M00  00190                                  276  00  00000  00000
D           13M00  00254                                  376  00  00000  00000
D           00M00  00100                                  144  00  00000  00000
D           01M00  00152                                  230  00  00000  00000
D           02M00  00190                                  276  00  00000  00000
D           03M00  00254                                  376  00  00000  00000
D           C1N00  00317                                  475  00  00000  00000
D           C0N00  00317                                  475  00  00000  00000
10M00   37  75701  75702  B      ALARM AND               144  37  75701  75702
10M01   MJ  00000  0             NORMAL EXIT             145  45  00000  00000
10M02   54  00M01  20017  BRB    ENTRY                   146  54  00145  20017
10M03   TU  A0000  00M11         P-1                     147  15  20000  00157
10M04   TU  A0000  01M09                                 150  15  20000  00241
10M05   AT  00015  A0000         P-2                     151  35  00017  20000
10M06   TU  A0000  00M21                                 152  15  20000  00171
10M07   AT  00015  A0000         P-3                     153  35  00017  20000
10M08   TU  A0000  00M19                                 154  15  20000  00167
10M09   TU  A0000  01M02                                 155  15  20000  00232
10M10   TU  A0000  01M03                                 156  15  20000  00233
10M11   TP  00000  A0000         SET                     157  11  00000  20000
10M12   TU  A0000  01M06          A                      160  15  20000  00236
10M13   TV  A0000  03M48            U                    161  16  20000  00456
10M14   AT  00015  A0000              X                   162  35  00017  20000
10M15   SS  00016  00015                 I                163  34  00020  00017
10M16   TU  A0000  03M48                   L              164  15  20000  00456
10M17   LA  A0000  00042                                 165  54  20000  00052
10M18   TV  A0000  01M06                                 166  16  20000  00236
10M19   TV  00000  02M50         SET F                   167  16  00000  00360
10M20   TN  00016  C0N10                                 170  13  00020  00507
10M21   TP  00000  A0000                                 171  11  00000  20000
10M22   TV  A0000  C0N06         SET N                   172  16  20000  00503
10M23   TU  A0000  C0N08         SET M                   173  15  20000  00505
10M24   AT  02M10  A0000         Y                       174  35  00310  20000
10M25   TV  A0000  01M13                                 175  16  20000  00245
10M26   TV  A0000  01M14                                 176  16  20000  00246
10M27   TV  A0000  03M02                                 177  16  20000  00400
10M28   54  C0N08  20071  BRB                             200  54  00505  20071
10M29   AT  C0N06  A0000                                 201  35  00503  20000
10M30   TV  A0000  C0N07         SET M-N                 202  16  20000  00504
10M31   AT  02M10  A0000         T                       203  35  00310  20000
10M32   TV  A0000  C0N01                                 204  16  20000  00476
10M33   TV  A0000  02M01                                 205  16  20000  00277
10M34   LA  A0000  00015                                 206  54  20000  00017
10M35   TU  A0000  02M06                                 207  15  20000  00304
10M36   TU  A0000  02M50                                 210  15  20000  00360
10M37   TU  A0000  02M11                                 211  15  20000  00311
10M38   TU  A0000  02M17                                 212  15  20000  00317
10M39   TP  C0N01  A0000                                 213  11  00476  20000
10M40   AT  C0N06  A0000         Z                       214  35  00503  20000
10M41   TV  A0000  03M16                                 215  16  20000  00416
10M42   TP  00021  Q0000                                 216  11  00025  10000
10M43   QS  C0N08  01M12                                 217  53  00505  00244
10M44   TP  C0N10  A0000         SET                     220  11  00507  20000
10M45   AT  C0N06  C0N09          N01                    221  35  00503  00506
10M46   54  C0N07  20017  BRB                             222  54  00504  20017
10M47   TU  A0000  01M25         M-N                     223  15  20000  00261
10M48   RA  00M01  C0N05         P-4 EXIT                224  21  00145  00502
10M49   TV  03M02  03M47                                 225  16  00400  00455
10M50   TU  02M62  01M17                                 226  15  00374  00251
10M51   TU  02M62  02M07                                 227  15  00374  00305
11M00   TV  02M50  01M01         SET 0 FOR               230  16  00360  00231
11M01   TP  00013  00000           INTERCHANGE           231  11  00015  00000
11M02   TV  00000  02M50         RESET                   232  16  00000  00360
11M03   54  00000  20017  BRB      TO                   233  54  00000  20017
11M04   TU  A0000  02M01           F ADDRESS            234  15  20000  00277
11M05   TP  C0N02  C0N11         SET SF INDEX            235  11  00477  00510
11M06   RJ  00000  00            TO AUX 1                236  37  00000  00000
11M07   TP  C0N07  00023         SET INDEX               237  11  00504  00027
11M08   TP  00021  Q0000                                 240  11  00025  10000
11M09   TP  00000  A0000         TEST FOR                241  11  00000  20000
11M10   SJ  01M11  01M16           INVERSION             242  46  00243  00250
11M11   RS  C0N11  00016                                 243  23  00510  00020
11M12   75  10000  01M14  BBR    AUGMENT                 244  75  10000  00246
11M13   TP  00013  00000           ROW OF                245  11  00015  00000
11M14   TP  C0N04  00000           UNIT                  246  11  00501  00000
11M15   RA  01M14  00016           MATRIX                247  21  00246  00020
11M16   TP  C0N04  A0000         CHECK IF ALL            250  11  00501  20000
11M17   TM  00000  00024           ELEMENTS IN           251  12  00000  00030
```

PX 71900-9-(125)

```
11M18   TJ  00024  00M00                ROW ARE                252  42  00030  00144
11M19   RA  01M17  00015                SCALED                 253  21  00251  00017
11M20   IJ  00023  01M16                CORRECTLY              254  41  00027  00250
11M21   RA  CON10  00016              ADVANCE AND              255  21  00507  00020
11M22   TP  CON10  00026                SET INDEX              256  11  00507  00032
11M23   QS  01M25  02M00                                       257  53  00261  00276
11M24   QS  01M25  02M45                                       260  53  00261  00353
11M25   MJ  00000  01M28                                       261  45  00000  00264
11M26   RS  02M00  00015                                       262  23  00276  00017
11M27   RS  02M45  00015                                       263  23  00353  00017
11M28   TU  02M00  02M49                                       264  15  00276  00357
11M29   TV  CON01  02M30              RESET                    265  16  00476  00334
11M30   TU  02M06  02M22                T                      266  15  00304  00324
11M31   RA  02M07  00015              ADVANCE                  267  21  00305  00017
11M32   TU  02M07  02M12                X                      270  15  00305  00312
11M33   TU  02M07  02M16                                       271  15  00305  00316
11M34   TU  02M07  02M21                                       272  15  00305  00323
11M35   TU  02M07  02M46                                       273  15  00305  00354
11M36   55  02M07  10025  BRB                                  274  55  00305  10025
11M37   TV  Q0000  02M26                                       275  16  10000  00330
12M00   75  30000  02M02  BBR         TRANSMIT ITH             276  75  30000  00300
12M01   TP  00000  0                    ROW TO ES              277  11  00000  00000
12M02   TV  03M26  02M21              SET FOR                  300  16  00430  00323
12M03   TV  03M26  02M22                INVERSION              301  16  00430  00324
12M04   TP  00013  00024                                       302  11  00015  00030
12M05   TP  00013  00028                                       303  11  00015  00034
12M06   TM  00000  00029              COMPARE LEAD             304  12  00000  00035
12M07   TM  00000  A0000                ELEMENTS               305  12  00000  20000
12M08   TJ  00029  02M16                                       306  42  00035  00316
12M09   ZJ  02M11  02M13                                       307  47  00311  00313
12M10   00  00000  CON12              CONSTANT                 310  00  00000  00511
12M11   MP  00000  CON03              R   INTER                311  71  00000  00500
12M12   DV  00000  00024                O                      312  73  00000  00030
12M13   RA  02M21  00016                  W   CHANGE           313  21  00323  00020
12M14   TN  CON04  00027                                       314  13  00501  00033
12M15   MJ  00000  02M20                                       315  45  00000  00322
12M16   54  00000  20043  B           NO ROW                   316  54  00000  20043
12M17   DV  00000  00024                INTER                  317  73  00000  00030
12M18   RA  02M22  00016                                       320  21  00324  00020
12M19   TP  00013  00027                  CHANGE               321  11  00015  00033
12M20   TN  00024  00024              OK                       322  13  00030  00030
12M21   TP  00000                     L                        323  11  00000  00000
12M22   TP  00000                       I                      324  11  00000  00000
12M23   54  00030  20043  BRB            N                     325  54  00036  20043
12M24   MA  00031  00024                 E                     326  72  00037  00030
12M25   MJ  00000  03M58                   A                   327  45  00000  00470
12M26   TP  B0000  000  0                    R                 330  11  30000  00000
12M27   TM  B0000  A0000                      L                331  12  30000  20000
12M28   TJ  00028  02M30                       Y               332  42  00034  00334
12M29   TP  A0000  00028                COMBINE                333  11  20000  00034
12M30   TP  00031  00000                                       334  11  00037  00000
12M31   RA  02M21  00015                R                      335  21  00323  00017
12M32   RA  02M22  00015                  O                    336  21  00324  00017
12M33   RA  02M30  00016                   W                   337  21  00334  00020
12M34   RA  02M26  00016                    S                  340  21  00330  00020
12M35   ST  CON01  A0000                                       341  36  00476  20000
12M36   ZJ  02M21  02M37                                       342  47  00323  00343
12M37   EJ  00028  02M47                R                      343  43  00034  00355
12M38   TV  00013  02M46                  E                    344  16  00015  00354
12M39   55  00028  10001  BRB              S                   345  55  00034  10001
12M40   QJ  02M44  02M41                   C                  346  44  00352  00347
12M41   QJ  02M47  02M42                     A                347  44  00355  00350
12M42   RA  02M46  00016                      L               350  21  00354  00020
12M43   QJ  02M45  02M42                       E              351  44  00353  00350
12M44   TV  03M56  02M46                R                      352  16  00466  00354
12M45   75  20000  02M47  BBR            O                     353  75  20000  00355
12M46   LA  00000                       W                      354  54  00000  00000
12M47   TP  00027  A0000              REPLACE ROW              355  11  00033  20000
12M48   SJ  02M49  02M51                ON DRUM IF             356  46  00357  00361
12M49   75  30000  02M51  BBR          INTERCHANGE              357  75  30000  00361
12M50   TP  00000                      TOOK PLACE               360  11  00000  00000
12M51   TP  00021  Q0000                                       361  11  00025  10000
12M52   TP  02M01  A0000                                       362  11  00277  20000
12M53   QA  02M00  02M01                                       363  52  00276  00277
12M54   LA  A0000  00057                                       364  54  20000  00071
12M55   TV  A0000  02M50                                       365  16  20000  00360
12M56   IJ  00026  01M26              I  TIMES                 366  41  00032  00262
12M57   IJ  CON09  00M50              N-1 TIMES                367  41  00506  00226
12M58   RS  CON07  CON06              SET M01                  370  23  00504  00503
12M59   RS  CON07  00016                FOR INDEX              371  23  00504  00020
```

```
12M60   54 CON06 20017 BRB
12M61   QS A0000 03M55
12M62   TP CON11 00029
12M63   TV 03M16 03M14
13M00   TP CON10 00026          SET INDEX
13M01   TU 02M01 03M14
13M02   TP CON04 00000          SET SCALE
13M03   TP 00021 Q0000
13M04   QS 00013 03M13
13M05   TP 00013 00028          COUNTERS TO
13M06   TP 00013 00027           ZERO
13M07   RA 00028 00015          ADVANCE
13M08   RA 03M13 00015           COUNT
13M09   RS 03M14 CON08
13M10   TU 03M14 03M16          TRANSFER
13M11   RS 03M14 00016           ROWS
13M12   RS 03M14 00028           OF UPPER
13M13   RP 30000 03M15           TRIANGULAR
13M14   TP 00000 0               MATRIX TO
13M15   RA 03M16 CON09           ES
13M16   TN 00000 00
13M17   54 03M16 20017 BRB
13M18   TU A0000 03M25
13M19   TV 03M02 03M25
13M20   TP 00027 00032          SET INDEX
13M21   TP 00013 A0000
13M22   MJ 00000 03M25
13M23   54 00030 20043 BRB                  B
13M24   CC 00031 00013                      A
13M25   MA 00000 0                          C
13M26   TP B0000 00030                      K
13M27   TP A0000 00031
13M28   TM 00030 A0000                      S
13M29   TJ CON04 03M31                      U
13M30   MJ 00000 03M54          RESCALE     B
13M31   RS 03M25 00017                      S
13M32   IJ 00032 03M23                      T
13M33   TU 03M25 03M36                      I
13M34   TV 03M25 03M44                      T
13M35   TU 03M25 03M44                      U
13M36   TM 00000 00024                      T
13M37   TN 00030 00030                      E
13M38   TN 00031 00031
13M39   TM 00030 A0000
13M40   TJ 00024 03M42
13M41   MJ 00000 03M54          RESCALE
13M42   54 00030 20043 BRB
13M43   CC 00031 00013
13M44   DV 00000
13M45   RA 00027 00016          ADVANCE
13M46   IJ 00026 03M07          NO1 TIMES
13M47   TP 00029 00000
13M48   RJ 00000 0               TO AUX 2
13M49   RA CON09 00015
13M50   IJ CON07 02M62          M-1 TIMES
13M51   MJ 00000 00M01          EXIT
13M52   11 CON00 75756 BRB      SET ALARM
13M53   MJ 00000 00M00           WORD
13M54   RS 00029 00016          RESCALE
13M55   RP 20000 03M57
13M56   LA CON13 00071
13M57   SJ 03M52 03M17
13M58   TP CON04 00032
13M59   SJ 03M60 03M61
13M60   TN CON04 00032
13M61   AT 00032 A0000
13M62   MJ 00000 02M26
C1N00   24 14061 33411 B        SINGUL
C1N01   TP B0000 000               C
C1N02   00 00000 00  42 B           O
C1N03   37 77777 77777 B             N
C1N04   20 00000 000  0 B          AND  S
C1N05   00 00000      3 B                T
C1N06   00 0                       TEMP   A
C1N07   00 0                                N
C1N08   00                         STORAGE  T
C1N09   00 0                                  S
C1N10   00
C1N11   00 0

372   54 00503 20017
373   53 20000 00465
374   11 00510 00035
375   16 00416 00414
376   11 00507 00032
377   15 00277 00414
400   11 00501 00000
401   11 00025 10000
402   53 00015 00413
403   11 00015 00034
404   11 00015 00033
405   21 00034 00017
406   21 00413 00017
407   23 00414 00505
410   15 00414 00416
411   23 00414 00020
412   23 00414 00034
413   75 30000 00415
414   11 00000 00000
415   21 00416 00506
416   13 00000 00000
417   54 00416 20017
420   15 20000 00427
421   16 00400 00427
422   11 00033 00040
423   11 00015 20000
424   45 00000 00427
425   54 00036 20043
426   27 00037 00015
427   72 00000 00000
430   11 30000 00036
431   11 20000 00037
432   12 00036 20000
433   42 00501 00435
434   45 00000 00464
435   23 00427 00021
436   41 00040 00425
437   15 00427 00442
440   16 00427 00452
441   15 00427 00452
442   12 00000 00030
443   13 00036 00036
444   13 00037 00037
445   12 00036 20000
446   42 00030 00450
447   45 00000 00464
450   54 00036 20043
451   27 00037 00015
452   73 00000 00000
453   21 00033 00020
454   41 00032 00405
455   11 00035 00000
456   37 00000 00000
457   21 00506 00017
460   41 00504 00374
461   45 00000 00145
462   11 00475 75756
463   45 00000 00144
464   23 00035 00020
465   75 20000 00467
466   54 00512 00107
467   46 00462 00417
470   11 00501 00040
471   46 00472 00473
472   13 00501 00040
473   35 00040 20000
474   45 00000 00330
475   24 14061 33411
476   11 30000 00000
477   00 00000 00042
500   37 77777 77777
501   20 00000 00000
502   00 00000 00003
503   00 00000 00000
504   00 00000 00000
505   00 00000 00000
506   00 00000 00000
507   00 00000 00000
510   00 00000 00000
```

February 13, 1956

THE RAMO-WOOLDRIDGE CORPORATION
Los Angeles 45, California

Digital Computing Center

STUDY OF MATRIX INVERSION ON THE ERA-1103 EMPLOYING ROUTINE MTI-0

by

Werner L. Frank and Phyllis Van Liew

Investigations have been carried on in the inversion of matrices of large order and/or matrices which are badly conditioned. The purpose of this has been twofold:

(1) To measure the sensitivity of the routine MTI-0 to ill conditioned matrices;

(2) To obtain some experimental experience relating to the effect of round-off for large order matrices.

(We define the condition number as the ratio of the absolute value of the largest eigenvalue to the smallest).

It is hoped that some conclusions will result which will answer the following questions:

(1) What is the relationship between condition number and resulting accuracy in the inverse?

(2) For moderately well conditioned matrices what order can be safely inverted and what accuracy can be expected?

The main problems considered in these investigations were the following matrices:

(a) Finite segment of a Hilbert Matrix (H_n) where $H_{ij} = \frac{1}{i+j-1}$ and its inverse T_n.

(b) Matrix associated with the solution of $y'' = -y$ (C_n) where $c_{ii} = 2$

$c_{ij} = -1$ if $|i-j| = 1$

$c_{ij} = 0$ all others

(c) A singular matrix (A) of order 8 studied in a paper "The Separation of Close Eigenvalues of a Real Symmetric Matrix" by Rosser, Lanczos, Hestenes and Karush.

Matrix (a) has the property of being extremely ill-conditioned for n as low as 4. The second matrix (b) enjoys a more moderate condition -- even though it is a function of the square of its order ($\sim \frac{4\ n^2}{2}$). In addition the inverses of both matrices are well known, the elements being given by closed algebraic forms.

More specifically, a comparison of the condition of H_n and C_n can be obtained from the following table (values are approximate):

n	C_n	H_n
2	3	16
4	10	15,514
10	40	$O(10^{12})$
39	638	?

Finally the matrix (c) is singular and its eigenvalues are known. The modified matrix $10^{-4}A + 10^{-n}I$ was studied for n = 1, 2,... , 10. This matrix has condition as given by

$$\frac{|\lambda_{max}| + 10^{-n}}{10^{-n}} = \frac{.1 + 10^{-n}}{10^{-n}} = 10^{n-1} + 1 = 10^{n-1}$$

The accuracy was checked by calculating the product of the input matrix and its computed inverse and comparing this quantity to the identity matrix. Tables 1 and 2 contain data associated with the matrices described above.

TABLE 1

Summary of Experiences on Hilbert Matrices and C_n

Matrix	Time (seconds) With Tape Output	Time (seconds) No Tape Output	Places of Accuracy (Rounded)	Condition
$T_2^{-1}.10^2$	1.90	.15	8	16
$T_3^{-1}.10^3$	3.65	.20	7	
$T_4^{-1}.10^5$	6.20	.45	5	15,514
$T_5^{-1}.10^6$	9.25	.70	3	
$T_6^{-1}.10^7$	13.10	1.00	3	
$T_7^{-1}.10^9$	17.50	1.45	1	
$T_8^{-1}.10^{10}$	22.7	2.1	0	
$H_2^{-1}.10$	1.90	.15	8	16
$H_3^{-1}.10$	3.70	.20	7	
$H_4^{-1}.10$	6.20	.40	5	15,514
$H_5^{-1}.10$	9.25	.60	4	
$H_6^{-1}.10$	13.05	1.00	2	
$H_7^{-1}.10$	17.45	1.40	1	
$H_8^{-1}.10$	22.6	1.9	0	
C_3			9	4
C_{15}			8	90
C_{27}	268	53	7	292
C_{39}	411		6	639
C_{99}	80 minutes (estimated)	30 minutes	5	4000

PX 71900-9-(125)

TABLE 2

The Matrix $10^{-4}A + 10^{-n}I = A_n'$

	Places of Accuracy in Identity Matrix (Rounded)	Condition
$10^{-4}A$	0	∞
A_{10}'	0	10^9
A_9'	1	10^8
A_8'	2	10^7
A_7'	3	10^6
A_6'	4	10^5
A_5'	5	10^4
A_4'	6	10^3
A_3'	7	10^2
A_2'	8	10
A_1'	9	1

UTILITY ROUTINE LIBRARY (MTØ)
FOR
1103 SERIAL 9 COMPUTER

The Utility Routine Library will normally be stored on MTØ. The Library consists of the following:

(1) Utility Routine Loader-Ø (URL-Ø)

This routine loads the Non-diagnostic Machine Test and transfers control to the test. An MT START automatically transfers control to URL-Ø; if (A_R) is set $\neq 0$, the machine test is by-passed and control is transferred to URL-1. The routine occupies blocks 1, 2 on MTØ.

(2) Non-Diagnostic Machine Test (Go-No-Go Test)

This routine performs nine tests each of machine commands and HSS, and four tests of the drum. A carriage return precedes the start of each part and a o is typed after each successful test. A command test failure is denoted by a printed c, a HSS test failure by a printed e, and a drum test failure by a printed d. After each failure the routine tries the same test over again with the same operands and continues to do so until it obtains a successful test. Upon completion, the routine transfers control to URL-1. The routine occupies blocks 3-32 on MTØ.

(3) Utility Routine Loader - 1 (URL-1)

This routine loads the Service Library in the drum, 70000-75777; it also clears HSS, groups 4, 5, 6 and 76000-77777 on the drum. A MJ-Ø is left in 00000. Upon completion, the computer is halted on a MS-Ø instruction with PAK set to the entry for the Ferranti Loading Routine. URL-1 occupies blocks 33, 34 on MTØ.

(4) Service Routine Library

This is a compilation of routines of general use. The Service Library occupies blocks 35-174 on MTØ.

(5) Regional Coding Routine (RECO)

This routine occupies blocks 175-225 on MTØ. See write up of this routine for detailed description.

(6) Library Routine

This routine occupies blocks 226-245 on MTØ. See write up of this routine for a detailed description.

(7) Dictionary of Subroutine Library

A list of indices for the subroutine stored in the Subroutine Library. This dictionary occupies blocks 246-305 on MTØ.

(8) Subroutine Library

A compilation of subroutines of general program use. The remaining blocks of MTØ are reserved for this library. See write up of Library Routine for list of Subroutine in the Library.

Loaders URL-2, and URL-3 are part of the Service Library and they effect the loading of RECO and the Library Routine respectively into HSS. The Library Routine has a built in loader which effects the transfer of the Dictionary and the Subroutine Library to HSS.

MTØ must be positioned to the dead space preceding the first block on tape before any use of the Utility Library can be made. To discourage inadvertent writing over the Utility Library, the MTØ WRITE switch is disabled; this switch in the DOWN position is the NORMAL condition of the computer.

GENERAL USE OF UTILITY ROUTINE LIBRARY

I. Normalize Computer Operation

This is an automatic operation which is designed to test the computer, transfer the Service Library to its drum storage, and clear HSS and the remainder of the drum.

Operating Instructions:

(1) Typewriter ON: MTØ positioned.

(2) MASTER CLEAR: All MJ and MS selects OFF.

(3) If the Non-diagnostic Machine Test is not desired, set $(A_R) \neq 0$. MT START.

(4) Computer halts on MSØ instruction with PAK set to 70001.

(5) Errors:

(a) Final Stop (57 77777 00000): This indicates that URL-Ø has not been transferred to HSS correctly. MASTER CLEAR, MT START for a re-transfer.

(b) Printed "T" and MS-Ø Stop: This indicates that the Non-Diagnostic Machine Test has not been transferred correctly. Push START for re-transfer of the test routine.

(c) Final Stop (57 77777 77777): This indicates that URL-1 has not been transferred to HSS correctly. MASTER CLEAR, MT START for a re-transfer.

(d) Printed "S" and MS-Ø Stop: This indicates that the Service Library has not been transferred correctly. Push START for a re-transfer of the Service Library.

NOTE: If repeated transfer errors occur and it is suspected that the Utility Library is incorrectly stored on MTØ, the machine operator should follow the procedure for loading the Utility Library on MTØ.

II. Program Assembly

Use of the assembly routines, RECO and the Library Routine, in the Utility Library is accomplished by entries 70010 and 70011 respectively in the Service Library. Entry at these points activates loaders which read the assembly routines into their operating storage locations in HSS. For detailed operating instructions, see descriptions of URL-2 and URL-3 in the Service Library.

III. Starting the Computer

There are three methods of starting the computer depending upon the amount of information stored on the drum.

(1) If the Service Library is stored correctly on the drum, one may use one of the routines in the library to load his program tape, and commence program operation and/or debugging.

(2) If the Service Library is not stored correctly on the drum, give an MT START for the Normalize Computer Operation (see description of same). This procedure is particularly advised when one suspects that the computer is malfunctioning.

(3) If the Utility Library is not available on MTØ and the Service Library is not on the drum, a bootstrap procedure must be performed by the operator (see "Procedure for Loading Utility Library on MTØ".)

PROCEDURE FOR LOADING UTILITY LIBRARY ON MTØ

If the Utility Library is not available on MTØ follow this procedure. This will work for a 16 interlace only.

(1) MASTER CLEAR

(2) Switch to ABNORMAL DRUM

(3) START

(4) After FINAL STOP, switch to NORMAL DRUM

(5) Set PAK=00000 and load tape I
The Ferranti Loading Routine is now on the drum.

(6) Set PAK=70001 and load tape II
The "Write Magnetic Tape Ø" routine is now in HSS.

(7) Position MTØ; turn MTØ WRITE Switch to UP position.

(8) MASTER CLEAR; MD START and load tape III.

The Utility Library is loaded onto MTØ.

(9) Set PAK=00077 START. This will cause a check sum of the information written an MTØ to be computed and compared with the sum computed from paper tape. If these two sums do not check a "tttt" is printed and computer stops. Push START for another comparison. If repeated check sum errors occur return to step (6). Skip step (7).

(10) After a successful loading of the Utility Library on MTØ, turn MTØ WRITE switch to DOWN position. Give an MT START to perform the Normalize Computer Operation.

PROGRAMMING AND OPERATION CONVENTIONS

I. Drum Image of HSS:

Drum cells 76000-77777 are reserved for the image of HSS. This image is used by most service routines as a temporary store for HSS while the service routine operates from HSS. The programmer is advised not to load into the image as this may result in incorrect loading of HSS. The programmer may use this part of drum storage as a temporary pool or work space during the operation of his program, but in so doing deprives himself of several facilities in the Service Library for program debugging.

II. Drum Storage for the Service Library

Drum cells 70000-75777 are reserved for the Service Library and are not, in general, available for program use. Loading programs into the range 70000-70037 deprives the programmer of all facilities of the Service Library, while loading into the range 70040-75777 may deprive him of only part of the Service Library.

III. Storage Used in Assembly of Subroutines

The remarks made in I and II concerning the reserved portions of group 7 on the drum apply also when assembling subroutines. The modified subroutines are stored at the specified relocation addresses, and then punched out, if the punch-out option is chosen.

IV. The MTØ unit is reserved for the Utility Library only. Any use of MTØ in an operating program must be brought to the machine operator's attention so that he may take necessary steps to preserve the Utility Library tape.

The NORMAL condition of the computer is indicated when the MTØ WRITE Switch is disabled (DOWN).

GENERAL PROCEDURE FOR COMPUTER OPERATION

The following items should be checked before going on the computer:

(1) List all memory used to determine if it is compatible with loading routines and Service Library.

(2) Do not try to load any tapes other than yellow or black. (Other tapes loaded at your own risk.)

(3) Keep tape off the floor.

(4) List library subroutines needed to determine if these routines are in the library.

(5) Determine which service routines will possibly be needed.

(6) Warm up the card equipment before using (at least one hour).

(7) Determine what peripheral equipment will be used.

When working with customers, the above information should be on hand before they arrive. For customers, determine the amount of assistance needed and/or wanted.

Before loading program tapes, it is suggested that the following procedure be followed:

(1) Check the ABNORMAL CONDITION panel.
If ABNORMAL light is on, check with the operator.

(2) Check the drum interlace if the drum is used in the program.

(3) Check the F_1 switch.

(4) Check the Field III switch in the card control unit if program uses card equipment. If the program is to use field III (Cols. 73-80), the switch should be in NORMAL Position.

(5) If any magnetic tape units are to be used in program, check to see if the units desired are switched to RUN and are properly positioned.

(6) If the High Speed Punch and/or Card equipment are not to be used in the program, turn them OFF.

(7) Turn Ferranti Reader OFF when not in use.

(8) If card equipment is to be used in the program be sure to clear both channels (read and punch) before giving a program START.

Upon leaving the computer, the following procedure is suggested:

(1) Check to see that all program tapes and output tapes are rewound and not left in baskets or on the tables.

(2) Clear the card equipment read, punch and receiving hoppers of all program input and output.

(3) If any information is stored on any of the magnetic tapes for future use, it is the programmers responsibility to see that these tapes are saved; otherwise it will be assumed that these tapes are usable for other programmers.

(4) Restore the Service Library to the drum.

CAUTION: It is advised as general practice never to Master Clear while any of the magnetic tape units are in motion. Such clearing drops all MT lockouts, and subsequent computer references to any moving tape will cause trouble.

SERVICE ROUTINE LIBRARY

Service Routine Entries

70000	Final Stop
70001	Ferranti Loading Routine (FRI-0)
70002	Flex Code Loading Routine (FLEXIE)
70003	Read
70004	Write
70005	Flex Dump
70006	Bioctal Dump
70007	-----
70010	Reco Loader (URL-2)
70011	Library Routine Loader (URL-3)
70012	-----
70013	Changed Word Post Mortem
70014	-----
70015	-----
70016	Single Breakpoint Stop
70017	Automatic Sampler
70020	Restore HSS from Image
70021	Punch Check Sum
70036	Common Exit
76000-77777	HSS Image-loading in this range results in incorrect loading of HSS.
70000-75777	Service Routine Library-loading in this range deprives one of the service routines.

PROGRAM ENTRIES TO SERVICE ROUTINES

The block of cells 70000-70037 is reserved for entries to the service routines. Cell 70036 is reserved as the common exit from those service routines which by their nature admit program entry and exit. For example, the use of FRI-0 as a subroutine would be effected by the instruction 37 70036 70001. All required parameter words must place in the appropriate registers before entry is made to the particular service routine by a Return Jump instruction. For example, the use of the Bioctal Dump would be effected by the following sequence:

```
                n:  11  (x)  10000
              n+1:  37 70036 70006
              n+2:  -- ----- -----
where, say, (x) =   00 00001 01777
```

CAUTION: Since the service routines each have only one entry, any inadvertent (or not) loading in the range 70000-70037 deprives one of all the service routines.

COMMENTS ON USE OF SERVICE LIBRARY

I. Paper Tape Preparation

a) Bioctal tapes should have double 7th level at the very end of the tape.

b) Flex code (absolute) program tapes should have at least one 7th level punch at the very end.

c) Flex code (RECO) program tapes should have an END. (car. ret.) or END (car. ret.) at the end of the tape.

d) Assembled RECO program tapes are suitable for direct input via FLEXIE. Such tapes may be converted to bioctal if so desired.

e) Flex dump tapes are suitable for reloading via Flexie. Be sure that a 7th level punch is present at the end of the tape.

f) Care must be taken in handling paper tapes to insure that they are kept clean. Foreign substances on tapes may cause improper loading into the computer.

II. Alarm Print

Many of the Subroutines contain references to the Convair Alarm Print. The location of this is at 75700-75777. The entries are:

37	75700	75701	main routine
37	75700	75702	subroutine

The alarm print in the service library is a modified version of the Convair routine. Program, constants and working space are in the range 75700-75777.

III. Flex Constant Pool

A group of commonly used Flexowriter codes are stored in the Service Library in the range 75757-75777. This pool is used by several Service routines; however, it is available for general program use. The pool consists of the following:

75757	00 00000 00037	0
75760	00 00000 00052	1
75761	00 00000 00074	2
75762	00 00000 00070	3
75763	00 00000 00064	4
75764	00 00000 00062	5

75765	00	00000	00066	6
75766	00	00000	00072	7
75767	00	00000	00060	8
75770	00	00000	00033	9
75771	00	00000	00045	Carriage Return
75772	00	00000	00004	Space
75773	00	00000	00057	Shift down
75774	00	00000	00047	Shift up
75775	00	00000	00051	Tab
75776	00	00000	00042	Period
75777	00	00000	00056	Minus.

IV. Responsibility of Restoring Service Library

It is the responsibility of any operator who damages the Service Library to restore the same to the drum before leaving the computer.

V. Loading Routines "Transfer Control" Option

The loading routines, FRI-0 and FLEXIE, now have a "transfer control" option. The following procedure will effect the transfer for either load routine.

(1) set program tape in reader

(2) MASTER CLEAR

(3) Set the computer on MAIN PULSE Ø.

(4) Manually insert the following into PCR

37 70036 70001(2)

(5) Set PAK= program start

(6) START.

PROCEDURE FOR LOADING SERVICE ROUTINE LIBRARY

I. Normal Operation

The Service Routine Library is normally stored on the drum, 70000-77777. Entry to the routines is achieved by setting PAK to one of the low-numbered drum address, 700XX.

II. MT Start

If the Service Library is damaged on the drum, select an MT Start. This will effect the following:

(1) Non-diagnostic machine test

(2) Transfer of Service Library to the drum

(3) Clearing of HSS and drum storage.

If (1) is not desired, set $(A_R) \neq 0$ before the MT Start.

III. Bootstrap

If the Service Library is not available on magnetic tape, follow this bootstrap procedure. This will work for a 16 interlace only.

(1) MASTER CLEAR

(2) Switch to ABNORMAL DRUM

(3) START

(4) After FINAL STOP, switch to NORMAL DRUM

(5) Set PAK=00000 and load tape I
The Ferranti Loading Routine is now on the drum.

(6) Set PAK=70001 and load tape II
The Service Library is now on the drum.

SERVICE ROUTINES

[Ferranti Loading Routine (FRI-0)]

The routine will load anywhere. Loading into HSS and 76000-77777 will result in incorrect loading of HSS. A sum check is made whenever the input tape contains an insert to 75202, 75203. (See Punch Check sum routine.)

Operating Instructions:

(1) Turn manual stop #1 OFF.

(2) Set PAK = 70001; START.

(3) Computer halts on 56 00000 70001 after completing read in. START to load another tape.

(4) Two consecutive seven-level punches in the trailer should be present. If these punches are not present, the following procedure may be used:

(a) FORCE STOP after the paper tape has passed through the reader.

(b) Set PAK = 00066 START. The last block of information read in is then stored in its proper location.

(5) Errors

(a) Machine prints "t" and halts. FRI-0 is not in HSS correctly and must be restored. START, transfer FRI-0 to HSS again. A second failure indicates that FRI-0 is not stored in the Service Library correctly. In this case, revert to Normalize Computer Operation.

(b) Machine prints "c" and halts. A check address has failed. STARTing ignores this error and routine proceeds as though error had not occurred. A check address failure should not be ignored as it is very likely that the paper tape is in error.

(c) Machine prints "m" and halts. Check sum has failed to agree with computed sum of data read in. START to ignore this error and to have routine continue read in.

[FLEXIE - A Flex Code Loading Routine]

This routine is designed to load Flex Code tape prepared on a Flexowriter in the conventional fashion for translating to bioctal. Flexie operates in the same fashion as the bioctal Ferranti Loading Routine. A sum check is made whenever the input tape contains an insert to 75202, 75203.

Operating Instructions:

(1) Set PAK = 70002; START

(2) Computer halts on 56 00000 70002 after completing read in. START to load another tape.

(3) At least one seven-level punch should be present in the trailer to stop the routine. If this punch is not present the following procedure may be used.

(a) FORCE STOP after the paper tape has passed through the reader.

(b) MASTER CLEAR; set PAK = 00022 START

(4) Errors:

(a) Machine prints "t" and halts. Flexie is not in HSS correctly and must be restored. START transfers Flexie to HSS again. A second failure indicates that Flexie is not stored in the Service Library correctly. In this case revert to the Normalize Computer Operation.

(b) Machine prints "c" and halts. A check address has failed. A START ignores this error and routine proceeds as if no error had occurred.

(c) Machine prints "m" and halts. A check sum has failed to agree with computed sum of data read in. START to ignore the error.

[Read]

This routine provides for the display of the contents of consecutive memory locations in Q, after an initial address is stated.

Operation Instructions:

(1) Enter initial address, a, in A_V.

(2) Set PAK = 70003

(3) START. (a) are displayed in Q. The address is automatically advanced for reading successive cells by repeated STARTs.

[Write]

This routine provides for the writing in consecutive memory locations after an initial address is stated.

Operating Instructions:

(1) Enter the address in A_V. Enter the contents for that address in Q.

(2) Set PAK = 70004 START.

(3) The address is automatically advanced for writing in successive cells by repeated STARTs.

[Flex Dump (FLEXO)]

This routine dumps the contents of consecutive storage cells on punched paper tape only. Automatic page editing is provided and every eighth address is given. The punched tape is suitable for re-loading via Flexie. A check sum is punched out at the end of the dump. (A_L), (A_R), (Q) are not restored or punched out. HSS is restored. This routine replaces RW-73 (Flex Dump).

Operating Instructions:

(1) Enter in Q_u the address of the first cell to be dumped.

Enter in Q_v the address of the last cell to be dumped.

If a seven-level punch stop code is desired at the end of the dump set Q_{35}=1.

(2) Turn ON the High Speed Punch.

(3) Set PAK = 70005 START.

(4) The machine halts on 56 00000 70005 providing a re-entry for another dump.

(5) Errors:

(a) Machine prints "t" and halts. Flexo is not in HSS correctly. START transfers Flexo to HSS again. A second failure indicates that Flexo is not stored in the Service Library correctly. In this case revert to the Normalize Computer Operation.

(b) Machine prints "p" and halts. An illegal parameter word has been set up in Q and is displayed there. Clear Q manually and insert correct parameter; START.

(6) Flexo dumps only one type of storage at a time, either HSS or drum. This routine does not use the 76000-77777 image, but uses an image 70400-70715 as a temporary store for part of HSS.

[Bioctal Dump]

This routine will dump onto paper tape in bioctal form the contents of any specified number of consecutive storage cells in HSS or the drum except 76000-77777. A check sum is automatically punched at the end of the dump. A double seven-level punch at the end of the tape is optional.

Operating Instructions:

(1) Enter in Q_u the address of the first cell to be dumped.

Enter in Q_v the address of the last cell to be dumped.

If a double seven-level stop code is to be punched following this dump, set $Q_{op} \neq 0$.

(2) Turn High Speed Punch ON.

(3) Set PAK = 70006 START.

(4) The stop at the end of the dump, 56 00000 70006, provides a re-entry for another dump. The contents of (A) and (Q) are not retained. HSS is restored at the end of the routine.

[RECO II LOADER (URL-2)]

This routine effects the transfer of RECO II from MTØ to HSS and transfers control to RECO after a successful sum check and rewind of MTØ.

Operating Instructions:

(1) Turn High Speed Punch and Flexowriter ON.

(2) Set program tape to be assembled in reader.

(3) Set PAK = 70010 START.

RECO is transferred from the Utility Library, a routine sum check performed and MTØ rewound. Control is then transferred to RECO and assembly begins. If no subroutines are assembled, the computer halts on 56 00000 00360 providing a re-entry to the assembly routine which is still in HSS in its correct form. If subroutines are assembled, RECO transfers control to URL-3 (Library Routine Loader) and the Computer finally halts on 56 00000 70011, providing a re-entry to URL-3. (See write-up of RECO II for more detailed description of tape preparation and handling.)

(4) Errors: Machine prints "a1", rewinds MTØ and halts. This means that RECO has been transferred incorrectly from the Utility Library on MTØ. START effects another transfer. Repeated errors indicate that RECO is not stored on MTØ correctly. In this case revert to the bootstrap procedure described in "Procedure for Loading Utility Library on MTØ".

[Library Routine Loader (URL-3)]

This routine effects the transfer of the Library Routine from MTØ to HSS and transfers control to the Library Routine upon completion of a successful sum check.

Operating Instructions:

(1) Turn High Speed Punch and Flexowriter ON. If no punch-out of assembled subroutines is desired set MJ1 select ON.

(2) Set Flex code tape of subroutine assembly information in reader. (NOTE: When assembling RECO tapes, the END psuedo code transfers control to URL-3; in this case the Flex tape of subroutine information must be spliced on the end of RECO tape.)

(3) Set PAK = 70011 START

The Library Routine is transferred from MTØ to HSS and a routine sum check performed. Control is then transferred to the Library Routine and assembly begins. Upon completion of assembly computer halts on 56 00000 70011 providing re-entry to the Library Routine Loader. (See write-up of Library Routine for more detailed description of tape preparation and handling.)

(4) Errors: Machine prints "a2", rewinds MTØ and halts. This means that the Library Routine has been transferred incorrectly from MTØ. START effects another transfer. Repeated errors indicate that the Library Routine is not stored on MTØ correctly. In this case revert to the bootstrap procedure described in "Procedure for Loading Utility Library on MTØ". For a list of errors encountered in assembly and the appropriate operating instructions for same, see write-up of Library Routine.

CAUTION: Do not assemble subroutines in the range 70000-77777.

[Changed Word Post Mortem]

This routine is designed to compare the contents of each word of HSS with its image at 76000-77777. The image contains (unless disturbed) the original contents of HSS as read into the computer. Those words in HSS which have been changed by the execution of the program are the only ones reported out.

Operating Instructions:

(1) Turn High Speed Punch ON.

(2) Set PAK = 70013 START.

(3) Computer halts on 56 00000 70013.

The following will be punched in Flex Code.

a) (Q)
b) (A_R)
c) (A_L)
d) Any changed word according to the following:

HSS word Image word HSS address

At the end of the routine, (A), (Q) and HSS are restored.

(4) This routine uses the cells 70600-70777 as temporary storage for part of HSS while the routine operates.

[Single Breakpoint Stop]

This routine permits one to select a single address of a program which one can run on high speed and stop before executing the instruction at that address. One may then sample the results of computation to date or step through several instructions.

Operating Instructions:

(1) Enter in Q_V the breakpoint address.

Enter in A_V the entry address for the program.

(2) Set PAK = 70016 START. The program will be executed up to the breakpoint at which time the computer will halt on 56 00000 70016, providing a re-entry for another breakpoint stop.

[Automatic Sampler (Sam-0)]

This routine provides for the printing or punching (in octal or decimal) the contents of any selected cells at selected check points. Output is suppressed for the first N_p times through the check point and after ($N_p + N_s$) times. The program which is being sampled is executed normally between check points. It is not necessary to provide for sampling while writing the program. The programmer stores in any available block of memory a list of information regarding check points, cells to be sampled, and scales.

A) Operating Instructions:

(1) Read in the program to be sampled. This is the unmodified problem program.

(2) Read in the "Sampling List Tape". See below for description of this tape.

(3) Set PAK = 70017 START. The routine sets up check points and transfers control to α which is contained in the sampling list.

B) Sampling List Tape (Flex or Bioctal).

This tape loads cell 73643 and the sampling list which contains a number of sublists, one for each check point. Each sublist contains all information necessary for sampling at one check point. This information may be stored in any convenient set of consecutive HSS or MD cells except 00100 through 00167.

The sampling list tape format is as follows:

	Fixed Storage	73643	XX	L_o	L_f
one complete sublist	Check point address	L_o	00	00000	c.p.
	Index word	L_1	00	N_p	N_s
	Parameter words	L_2	0a	M	s
		.	.	.	.
		.	.	.	.
		.	0a	M	s
	End word	.	70	00000	00000
	End word	.	70	00000	00000
		.	00	00000	c.p.
		.	00	N_p	N_s
		.	0a	M	s
		.	.	.	.
		.	.	.	.
		.	70	00000	00000
		L_f	70	00000	α

(1) Fixed storage - The word XX L_o L_f read into 73643, gives the address of the first cell (L_o) and the last cell (L_f) of the sampling list. Printing or punching is specified by XX, 61 for printing and 63 for punching.

(2) Check point address - The word 00 00000 c.p. gives the check point address, c.p. Sampling occurs before execution of the instruction at c.p.

PX 71900-9-(126)

(3) Index word - The word 00 N_p N_s gives two 5-octal-digit numbers, N_p, signifying the number of times the check point is to be passed before sampling starts, and N_s, the number of times sampling is to occur at the check point.

(4) Parameter words - These are of the form 0a M s where the first octal digit is always zero. The second octal digit, a, takes on the values

0 for octal output
1 for decimal output.

If a ≥ 1, the parameter word is ignored. The u-portion of each parameter word contains the octal address, M, of a cell whose contents are to be sampled. If M = 20001, (A_L) is sampled. If M is not a machine address the parameter word is ignored. The v-portion of each parameter word contains the binary scale factor, s, of the contents of M. $0 \leq s < 70$. If $s \geq 70$, "2 small" is printed.

(5) End words - The last two words of each sublist are of the form 70 00000 00000 with the exception of the second end word of the last sublist (L_f), which is 70 00000 α . SAM-0 jumps to α after setting up check points on a 70017 start.

C) Output

Shown below is an example of sampler output where the check point address was 00303.

```
00303
00075        12 34567 12340
00076        77 03124 65432
00100         1.23456789017
00101      -321.098632812
00102       993059913.000
00103         0.43210987653
20001        14 00000 00000
10000        37 37373 73737
```

D) Restrictions

The word initially stored at a check point must be an instruction; it must not be a repeat command or a repeated instruction and it may not be written into or out of at any time during the course of the program.

PX 71900-9-(126)

[Restore HSS from Image]

This routine provides for an automatic transfer of the image, 76000-77777, to HSS.

Operating Instructions:

(1) Set PAK = 70020 START.

(2) Computer halts on 56 00000 70020.

[Punch Check Sum]

This routine is to be used immediately after loading a program tape by means of the Ferranti Loading Routine (FRI-0). It punches a trailer containing the sum of the loaded program with proper insert and check addresses. This trailer can then be merged with the program tape to provide a check sum upon subsequent loadings.

Operating Instructions:

(1) Load the program tape. There must be no double seven-level punches in the program tape. Thus after loading with FRI-0, the computer will hang up on an External Read• FORCE STOP the computer.

(2) MASTER CLEAR; set PAK = 70021

(3) START. The following punched tape will be produced:

IA 75202
CHECK
SUM
CA 75204
Double seven-level punches.

The program will be properly loaded at the conclusion of this routine.

REGIONAL CODING ROUTINE II (RECO II)

Introduction

RECO is a one-pass assembly routine designed to translate an 1103 program coded in symbolic, regional address form to 1103 machine language.

Paper tape punched in Flexowriter code is the input-output medium for RECO. Input is the tape of RECO-coded program to be assembled. Except for the input of decimal numbers, all numbering in this program is recorded in the octal system. Output is the tape of the assembled program in octal notation. The output is translatable to bioctal form via the ERA converter, or may be loaded directly into the computer using the ERA "Flexie" load routine. A listing of the assembled program may be readily obtained on a Flexowriter.

RECO offers considerable flexibility in recording 1103 words. The 1103 symbolic and numeric operation codes, as well as those of the Interpretive System are available for use. Addresses may be recorded in either absolute or relative form. Relative addresses use a two-letter regional address code; in addition, the one letter regional address codes A and Q refer automatically to the accumulator and Q-register. A maximum of 100 distinct regional address codes is available in a RECO coded program, and the pseudo-instructions assigning absolute addresses to each of these codes precede the main program on the input tape.

Provision is made for the direct input of program constants written in decimal notation. A decimal number, N, is automatically converted by RECO to its signed 36-bit equivalent in fixed point, or floating point, form. The floating point form offers full range conversion, the range being roughly,

$$10^{-38} \leq |N| \leq 10^{38}$$

All decimal numbers are identified as such by a RECO pseudo-operation code.

The punching of insert and check addresses on the output tape is obtained by using pseudo-instructions. A check sum of assembled 1103 words is automatically punched out with each check address instruction. A pseudo-instruction is available whereby the programmer can insert a MJOv instruction in addresses 00000 and 40000; this pseudo-instruction permits him to begin operation of his program with an MD start.

Two types of terminating instructions are available. The first of these punches out trailer and halts the computer. The second punches out trailer and transfers control to the ERA Library Routine. Thus, if the programmer desires the assembly of certain subroutines for his program he must record the proper terminating command followed by a list of subroutine indices at the end of his input tape. This option permits continuous assembly of the complete program. The total output in this case would consist of two parts on one continuous tape: (1) Assembled main program; (2) Assembled subroutines.

Several types of program errors are detected by RECO. The type of

error and its location on the input tape are printed out on the Flexowriter. All RECO output ceases as the first program error is detected, and the RECO input tape is processed from that point on for errors only. This error detection is limited in the sense that only the first error of a program word is recorded.

A detail description of RECO programming is given below. Part I discusses the writing of regular 1103 words in symbolic form. Part II lists the seven RECO pseudo-instructions. Part III describes input-output format. Part IV describes RECO error detection. Part V lists RECO operating instructions. Figure 1 of Part III is an example of a RECO program. Figure 2 displays a listing of the assembled program given in Figure 1.

I. 1103 Words

A. Address Portion

Addresses may be recorded in either absolute or relative form. RECO requires that at least one digit be recorded for any absolute address. Relative addresses appear as $a_1\ a_2$ B, where ($a_1\ a_2$) is a two-letter code denoting any one of a maximum of 100 (decimal) addressable regions, and where (B) designates the sequence number of the address within the region ($a_1\ a_2$). The only restriction on the choice of letters in the regional address code is that $a_1 \neq$ A or Q. (B) may be zero, in which case only ($a_1\ a_2$) need be recorded.

The regional portion, ($a_1\ a_2$), of a relative address is assigned its absolute address value, δ, by means of the RECO pseudo-instruction RE $a_1\ a_2\ \delta$. All such assignments should be recorded at the beginning of the program tape.

RECO requires that an entry be made for each address portion of every 1103 instruction and octal constant. For an instruction in which an address portion is a "j", "jn", or "jk" number (e.g. RPjnv), all five octal digits must be recorded; if j = 0, one need only record 0, n, or k.

B. Operation Portion

Operation codes are either the two digits or the two letters of the 1103 instruction repertoire. The numerical operation codes extend from 00 to 77. In addition, the alphabetical,numeric, or alpha-numeric operation codes used in the 1103 Interpretive System are recognized by RECO as legal operations. However, the operation code for the Interpretive Replace Multiply is restricted in form to either IP70 or 1470.

Those 1103 words which form octal constants with absolute or relative address portions, must have at least one and no more than two digits recorded in the operation portion of the word.

C. Decimal Numbers

All decimal numbers are converted by RECO to signed 36-bit equivalents. Decimal numbers must be identified as such by the pseudo-operation code

DE; these are the only program words for which RECO accepts numbering in the decimal system.

There are two forms of decimal number words:

(1) Fixed Point DE $\pm$ i.$f_1f_2 \ldots f_9f_{10}$ P S, where $1 \le i \le 9$, $0 \le f_j \le 9$, $-99 \le P \le 99$, such that $0 \le \log_2 \left| \pm i.f_1f_2 \ldots f_9f_{10} \times 10^P \right| + S < 35$. "S" is the binary scale factor applied to $\left| \pm i.f_1f_2 \ldots f_9 f_{10} \times 10^P \right|$. "P" is the power of ten.

(2) Floating Point DE $\pm$ i.$f_1f_2 \ldots f_9f_{10}$ P F, where $1 \le i \le 9$, $0 \le f_j \le 9$, $-39 < P < 39$. "F" is the code letter denoting floating point conversion to packed (28-8) form with biased characteristic.

The range on a floating point decimal number, N, is roughly $10^{-38} \le |N| \le 10^{38}$.

The plus sign (+) is optional on all signed quantities in the decimal number. The decimal point and some value for P and S must always be recorded.

L. Subroutine Assembly

RECO does not assemble library subroutines directly; rather, it provides the option of transferring control at the end of a RECO assembly to another program which assembles subroutines. The use of this option terminates all RECO assembly and punches out those subroutines listed on the input tape. This list of subroutines must be prefaced by the RECO instruction END and must comply with the input format of the ERA Library Routine.

The output tape resulting from use of this subroutine option is one continuous flex code tape consisting of two parts: (1) Assembled main program; (2) assembled subroutines.

It is imperative that all, if any, subroutine assembly be the final step in a complete program assembly. The reason for this is that the use of the subroutine option in effect destroys all previous RE $\alpha_1\alpha_2\delta$ instructions, and of course the information contained in these instructions is necessary for proper RECO assembly.

II. RECO Instructions or Words

A. RE $\alpha_1\ \alpha_2\ \delta$ "Assign the region $\alpha_1\ \alpha_2$ the value δ " This instruction is described in I-A above.

B. DE $\pm i.f_1f_2 \ldots f_9f_{10}$ P S(F) "Decimal Number Fixed (Floating)" This instruction is described in I-C above.

C. IA $\alpha_1\ \alpha_2$B "Insert address at $\alpha_1\ \alpha_2$B" This instruction causes a six-inch leader to be punched out followed by the translated insert address, $\alpha_1\ \alpha_2$B.

D. CA $\alpha_1\ \alpha_2$B "Check address at $\alpha_1\ \alpha_2$B" This instruction causes the translated check address, $\alpha_1\ \alpha_2$B, to be punched out, followed by a punch out of a check sum of all 1103 words since last IA instruction. This check sum is inserted at machine addresses 75202 and 75203 on the output tape.

E. START $\alpha_1\ \alpha_2$B. "Start program at $\alpha_1\ \alpha_2$B" This instruction causes a translated MJO ($\alpha_1\ \alpha_2$B) instruction to be inserted at addresses 00000 and 40000 on the assembled program tape. The correct addresses and check sums are automatically punched out. This instruction is optional.

F. END. "End RECO program assembly" This instruction causes a 10 inch trailer, followed by two periods and a seventh level to be punched out, before halting the computer.

G. END "End RECO program assembly, begin subroutine Assembly"
This instruction causes a 10 inch trailer to be punched out, and transfers control to the subroutine assembly program as described in I-E above.

III. Input-Output Format

Figures 1 and 2 are illustrations of the horizontal and vertical input-output formats. The format exhibited in Figure 1 is strongly recommended for RECO programming. Every 1103 and RECO word must be terminated by at least one vertical space (i.e. carriage return), and the distinct portions of any word must be separated by at least one horizontal space (i.e. space or tab). RECO ignores all leader, shift up, shift down, and delete flex codes. In particular, flex characters for "1" and 1 (one), and o and 0(zero) are not interchangeable.

Figure 1

```
RE  BC10
RE  ZA40
RE  TQ100
RE  RS50000

START  BC

IA  BC
TP  TQ10  A42
12  TQ6  Q
SP  TQ14  17
EF  0  TQ13
77  10000  20000
LA  TQ5  A17
DV  TQ6  TQ14
IPMP  ZA  ZA5
RP  30005  BC12
TP  RS  ZA
PR  0  BC
MS  30000  BC14
RS  111  TQ12
1473  TQ10  TQ14
RJ  RS5  RS
MJ  0  ZA
CA  BC20

IA  TQ5
DE  +2.147483648  9  F
DE  -7.3786976294  19  -36
DE  2.9103830456  -11  35
DE  3.14159  5  F
DE  1.0  10  0
DE  3.162278  -38  F
0  7  0
0  0  0
DE  1.7014118  38  F
CA  TQ16


END.
```

Figure 2

```
.000000.040000.
45 00000 00010.
00000004.0001.
000000.075202.
00 00000 00000.

45 00000 00010.
00000007.5204.


.000000.000000.
45 00000 00010.
00000000.0001.
000000.075202.
00 00000 00000.

45 00000 00010.
00000007.5204.
```

```
bc
.000000.000010.
11 00110 20042.
12 00106 10000.
31 00114 00017.
17 00000 00113.

77 10000 20000.
54 00105 20017.
73 00106 00114.
14 71004 00045.


75 30005 00022.
11 50000 00040.
61 00000 00010.
56 30000 00024.

23 00111 00112.
14 73011 00114.
37 50005 50000.
45 00000 00040.


00000000.0030.
000000.075202.
00 00000 00007.
70 74706 41000.

00000007.5204.
```

Figure 2 (continued)

```
tq+
.000000.000105.
20 00000 00240.
76 77777 77777.
00 00000 00001.
23 13136 00223.

11 24027 62000.
25 41273 44004.
00 00007 00000.
00 00000 00000.


37 77777 76777.
00000000.0116.
000000.075202.
00 00000 00002.

41 00470 25466.
00000007.5204.


..
```

Format of output listing is exhibited in Figure 2. This listing of an assembled program should prove to be useful in program debugging. The output tape is translatable to bioctal form or may be loaded directly into the computer via the ERA "Flexie" load routine.

Flex Code stops are automatically punched by RECO after each check sum check address to facilitate bioctal translation or listing.

IV. Error Detection

RECO detects a limited number of program errors. These error types and their respective tags are as follows:

OC Illegal operation code

RC Illegal regional address code (i.e. A a_2 or Q a_2)

CA Check address failure

RA Unassigned relative address region

AO Relative address exceeds the value 77777

OD Illegal octal digit flex code

FE Decimal point missing in decimal number

FO Number of fractional digits (f_j) > 10 in decimal number

UN Decimal number underflow (i.e. loss of significant digits)

DD Illegal decimal digit flex code

OV Decimal number overflow (i.e. loss of significant digits)

RECO detects only the first error in a programmed 1103 or RECO word. This detection does not apply to any errors occurring in the listing of subroutine indices at the end of the input tape; errors in this portion of the input tape are detected by the subroutine program.

As RECO encounters a program error, it prints out the error tag followed by two decimal numbers, a group count and a word count, which locate the error on the input tape. Group 0 includes all RECO words prior to the first programmed insert address instruction, which is word 1 of group 1. Each succeeding insert address begins a new group.

All output ceases as RECO detects the first program error, and input tape is processed for errors only from that point on. This termination of output does not include that output resulting from subroutine assembly.

If the number of RECO instructions RE a_1 $a_2\delta$ exceeds 100 (decimal) the computer will halt with an SCC fault. RECO will not assemble such a program correctly.

Improper horizontal and vertical spacing is also indirectly detected by RECO as one of the error types listed above.

V. Operating Instructions

1. Flexowriter and high speed punch on.

2. Load input tape in Ferranti reader and position the tape several blank frames ahead of first character.

3. Set all MJ and MS selects off; master clear; set PAK to 70010; START. RECO is brought into HSS from magnetic tape and assembly begins.

   a. If the instruction END. is recorded at the end of the input tape, the computer will halt on a 56 00000 00360 instruction. If END. is omitted, the computer must be force stopped after input tape has passed through the reader.

   b. If the subroutine option, END, is on the input tape and a period is recorded after the last subroutine index, the computer will halt on a 56 00000 70011 instruction. If the period is omitted, the computer must be force stopped after input tape has passed through the reader.

4. To assemble a second program, and case (3)-a applies to the preceding program assembly, position second tape in reader, set PAK to 00360, and START.

5. To assemble a second input tape making direct use of RE $\alpha_1 \alpha_2 \delta$ instructions from the preceding tape and case (3)-a applies, position the second tape in reader, set PAK to 00364 and START.

RR-126

```
00001  00 00000 02031
00002  00 00000 02012
00003  00 00000 01215
00004  00 00000 02426
00005  00 00000 02217
00006  00 00000 00730
00007  00 00000 00715
00010  00 00000 00000
00011  00 00000 02307
00012  00 00000 03007
00013  00 00000 03107
00014  00 00000 01207
00015  00 00000 01534
00016  00 00000 00000
00017  00 00000 01512
00020  00 00000 00000
00021  00 00000 02624
00022  00 00000 00724
00023  00 00000 01135
00024  00 00000 01130
00025  00 00000 03524
00026  00 00000 03530
00027  00 00000 03501
00030  00 00000 00000
00031  00 00000 02132
00032  00 00000 02432
```

Flex codes for computer instruction repertoire.

addresses 00001 thru 00100

PX 71900-9-(126)

```
00033  00 00000 00732
00034  00 00000 03532
00035  00 00000 02032
00036  00 00000 00132
00037  00 00000 01432
00040  00 00000 00000
00041  00 00000 01232
00042  00 00000 02401
00043  00 00000 03001
00044  00 00000 02424
00045  00 00000 02406
00046  00 00000 02430
00047  00 00000 02415
00050  00 00000 00000
00051  00 00000 01616
00052  00 00000 00000
00053  00 00000 00707
00054  00 00000 01507
00055  00 00000 01224
00056  00 00000 01101
00057  00 00000 01230
00060  00 00000 00000
00061  00 00000 02026
00062  00 00000 00117
00063  00 00000 00134
```

```
00064  00 00000 01415
00065  00 00000 00106
00066  00 00000 00107
00067  00 00000 00115
00070  00 00000 00000
00071  00 00000 00000
00072  00 00000 00000
00073  00 00000 00000
00074  00 00000 00000
00075  00 00000 00000
00076  00 00000 00000
00077  00 00000 00000
00100  00 00000 00000
00101  00 00000 07272
00102  00 00000 07266
00103  00 00000 07262
00104  00 00000 07264
00105  00 00000 07270
00106  00 00000 07274
00107  00 00000 07252
00110  00 00000 07237
00111  00 00000 06672
00112  00 00000 06666
00113  00 00000 06662
00114  00 00000 06664
00115  00 00000 06670
00116  00 00000 06674
00117  00 00000 06652
00120  00 00000 06637
00121  00 00000 06272
```

Flex codes for
computer numeric
operations.
addresses 00101 thru 00200

00122 00 00000 06266

00123 00 00000 06262

00124 00 00000 06264

00125 00 00000 06270

00126 00 00000 06274

00127 00 00000 06252

00130 00 00000 06257

00131 00 00000 06472

00132 00 00000 06466

00133 00 00000 06462

00134 00 00000 06464

00135 00 00000 06470

00136 00 00000 06474

00137 00 00000 06452

00140 00 00000 06457

00141 00 00000 07072

00142 00 00000 07066

00143 00 00000 07062

00144 00 00000 07064

00145 00 00000 07070

00146 00 00000 07074

00147 00 00000 07052

00150 00 00000 07057

```
00151  00 00000 07472
00152  00 00000 07466
00153  00 00000 07462
00154  00 00000 07464
00155  00 00000 07470
00156  00 00000 07474
00157  00 00000 07452
00160  00 00000 07437
00161  00 00000 05272
00162  00 00000 05266
00163  00 00000 05262
00164  00 00000 05264
00165  00 00000 05270
00166  00 00000 05274
00167  00 00000 05252
00170  00 00000 05237
00171  00 00000 03772
00172  00 00000 03766
00173  00 00000 03762
00174  00 00000 03764
00175  00 00000 03770
00176  00 00000 03774
00177  00 00000 03752
00200  00 00000 03737
```

```
00201  00 00000 01220
00202  00 00000 66242
00203  00 24013 01201
00204  00 00000 02220
00205  00 00000 01430
00206  00 00000 01630
00207  00 00141 50000
00210  00 00526 40000
00211  00 00000 00000
00212  00 00000 00077
00213  00 00000 00047
00214  00 00000 00057
00215  00 00000 00004
00216  00 00000 00045
00217  00 00000 00051
00220  00 00000 00033
00221  00 00000 00060
00222  00 00000 00072
00223  00 00000 00066
00224  00 00000 00062
00225  00 00000 00064
00226  00 00000 00070
00227  00 00000 00074
00230  00 00000 00052
00231  00 00000 00037
00232  00 00000 00030
00233  00 00000 00035
00234  00 00002 00622
```

Program constants.

Addresses 00201 thru 00357

Program temporary storage

at addresses 01430 thru 01467

```
00235  00 00777 70000
00236  00 00000 01400
00237  00 00000 07777
00240  00 00000 00017
00241  00 00000 00000
00242  00 00000 00473
00243  00 00000 00013
00244  00 00000 20000
00245  00 00000 10000
00246  00 00000 00001
00247  00 00000 01634
00250  67 67600 00000
00251  00 00000 00726
00252  00 00000 00667
00253  00 00000 00007
00254  63 00000 00255
00255  00 00000 00037
00256  00 00000 00052
00257  00 00000 00074
00260  00 00000 00070
00261  00 00000 00064
00262  00 00000 00062
00263  00 00000 00066
00264  00 00000 00072
00265  00 00000 01247
00266  61 00000 00231
00267  00 00000 00042
```

```
00270  00 00000 00004
00271  00 00000 00003
00272  00 00000 75202
00273  00 00000 75204
00274  00 00000 00403
00275  00 00000 00742
00276  00 00000 00532
00277  45 00000 00000
00300  00 00000 00763
00301  00 00000 40000
00302  00 00000 40001
00303  00 00000 00077
00304  00 00000 00002
00305  00 20000 00000
00306  00 00001 00000
00307  00 00000 77777
00310  00 00000 00777
00311  00 00000 01273
00312  00 00007 00000
00313  00 00000 01470
00314  00 00000 00565
00315  00 00000 01250
00316  11 24027 62000
```

```
00317  00 73465 45000
00320  00 05753 60400
00321  00 00461 13200
00322  00 00036 41100
00323  00 00003 03240
00324  00 00000 23420
00325  00 00000 01750
00326  00 00000 00144
00327  00 00000 00012
00330  00 00000 00001
00331  71 01446 00330
00332  00 00000 01445
00333  00 00000 01457
00334  00 00000 00026
00335  00 00000 00056
00336  00 00000 00054
00337  00 00000 00043
00340  00 00000 00110
00341  77 77777 77734
00342  20 00000 00000
00343  77 77777 77735
00344  00 00000 00105
00345  00 00000 00200
00346  00 00000 00377
00347  00 00000 00001
00350  00 00000 01254
00351  00 00000 01253
```

```
00352  45 00000 00441
00353  45 00000 00463
00354  45 00000 00665
00355  45 00000 00723
00356  45 00000 00762
00357  00 00000 00010
00360  11 00305 01437
00361  16 00313 00646
00362  16 00247 00651
00363  11 00277 00000
00364  17 00000 00312
00365  76 00000 20000
00366  16 00276 00531
00367  16 00252 00666
00370  16 00251 00725
00371  16 00300 00762
00372  16 00265 01246
00373  16 00351 01252
00374  11 00246 00347
00375  11 00241 01463
00376  11 00241 01464
00377  11 00241 01434
00400  11 00241 01435
00401  11 00241 01436
00402  45 00000 00403
00403  37 00604 00600
00404  11 20000 01430
00405  37 00615 00605
```

PROGRAM ENTRY

AND

PRESET ROUTINE .

ADDRESSES -

00360 THRU 00402

```
00406  11 01430 20000
00407  75 20100 00411
00410  43 00001 00432
00411  75 20100 00413
00412  43 00101 00432
00413  75 20010 00415
00414  43 00222 00432
00415  43 00201 00656
00416  43 00202 01245
00417  43 00204 01004
00420  43 00205 00660
00421  43 00206 00756
00422  43 00203 00760
00423  43 00234 01251
00424  11 00235 10000
00425  51 20000 20000
00426  43 00207 00445
00427  43 00210 00445
00430  21 01464 00246
00431  45 00000 01266
00432  21 01464 00246
00433  31 10000 00036
00434  11 20000 01432
00435  37 00472 00465
00436  54 20000 00017
```

MAIN CONTROL

ROUTINE.

ADDRESSES -

00403 THRU 00464

PX 71900-9-(126)

```
00437  15 20000 01432
00440  37 00472 00465
00441  16 20000 01432
00442  11 01432 10000
00443  37 00565 00530
00444  45 00000 00403
00445  21 01464 00246
00446  11 00237 10000
00447  51 01430 20000
00450  75 20100 00452
00451  43 00001 00454
00452  75 20100 00431
00453  43 00101 00454
00454  51 00212 20000
00455  32 00236 00030
00456  11 20000 01432
00457  37 00472 00465
00460  54 20000 00014
00461  35 01432 01432
00462  37 00472 00465
00463  35 01432 10000
00464  45 00000 00443
00465  37 00604 00600
00466  37 00636 00630
00467  11 20000 01431
00470  37 00627 00616
00471  11 01431 20000
```

```
00472  45 00000 00000
00473  43 00232 00521
00474  43 00233 00526
00475  54 20000 00006
00476  11 20000 01433
00477  37 00577 00571
00500  35 01433 01447
00501  15 01437 00502
00502  75 20000 01271
00503  43 01470 00504
00504  31 00502 00071
00505  16 20000 01433
00506  21 10000 00246
00507  23 01433 10000
00510  21 01433 00247
00511  16 01433 00515
00512  11 00241 01451
00513  37 00627 00616
00514  11 01431 01450
00515  21 01431 01634
00516  11 00307 20000
00517  42 01431 01272
00520  45 00000 00471
00521  11 00244 01433
00522  11 00241 01431
00523  37 00627 00616
00524  21 01431 01433
00525  45 00000 00516
```

U AND V

ADDRESS

DECODING

ADDRESSES -

00465 THRU 00527

PX 71900-9-(126)

```
00526  11 00245 01433
00527  45 00000 00522
00530  21 01434 00246
00531  45 00000 00532
00532  31 01435 00044
00533  32 01436 00000
00534  32 10000 00000
00535  11 20000 01436
00536  54 20000 00044
00537  11 20000 01435
00540  11 00243 01433
00541  11 00250 01432
00542  11 10000 01440
00543  11 01432 10000
00544  44 00550 00545
00545  11 10000 01432
00546  63 00000 00215
00547  45 00000 00543
00550  11 10000 01432
00551  55 01440 00003
00552  51 00253 20000
00553  35 00254 00554
00554  63 00000 00000
00555  41 01433 00543
00556  63 00000 00267
00557  63 00000 00216
```

OUTPUT (PUNCH)

ROUTINE

ADDRESSES-

00530 THRU 00570

```
00560  41 01466 00562
00561  45 00000 00566
00562  41 01465 00565
00563  63 00000 00216          READ,
00564  11 00271 01465          DECODE,
00565  45 00000 00000          THROWAWAY,
00566  63 00000 00216          ROUTINES.
00567  11 00853 01466          ADDRESSES -
00570  45 00000 00563          00571 THRU 00636
00571  17 00000 00312
00572  76 00000 20000
00573  43 00211 00571
00574  43 00212 00571
00575  43 00213 00571
00576  43 00214 00571
00577  45 00000 00000
00600  37 00577 00571
00601  43 00215 00600
00602  43 00216 00600
00603  43 00217 00600
00604  45 00000 00000
00605  37 00577 00571
00606  43 00215 00615
00607  43 00216 00615
00610  43 00217 00615
00611  55 01430 00006
00612  32 10000 00000
00613  11 20000 01430
```

```
00614  45 00000 00605
00615  45 00000 00000
00616  37 00577 00571
00617  43 00215 00627
00620  43 00216 00627
00621  43 00217 00627
00622  37 00636 00632
00623  55 01431 00005
00624  32 01431 00000
00625  11 20000 01431    SETUP DICTIONARY
00626  45 00000 00616    OF REGIONAL ASSIGNMENTS
00627  45 00000 00000    AND MACHINE ADDRESS
00630  16 00242 00633    EQUIVALENTS ROUTINE.
00631  45 00000 00633    ADDRESSES -
00632  16 00311 00633    00637 THRU 00657
00633  75 20010 00000
00634  43 00222 00635
00635  51 00240 20000
00636  45 00000 00000
00637  37 00604 00600
00640  43 00232 01267
00641  43 00233 01267
00642  54 20000 00006
00643  11 20000 01431
00644  37 00577 00571
00645  32 01431 00000
```

```
00646  11 20000 01470
00647  11 00241 01431
00650  37 00627 00616
00651  11 01431 01634
00652  21 00646 00246
00653  21 00651 00246
00654  21 01457 00306
00655  45 00000 00403
00656  21 01464 00246
00657  45 00000 00637
00660  21 01463 00246
00661  11 00246 01464
00662  11 00241 01447
00663  11 00241 01450
00664  37 00472 00465
00665  11 20000 01434
00666  45 00000 00667
00667  75 00074 00671
00670  63 00000 00241
00671  11 01447 20000
00672  47 00673 00701
00673  55 01447 10036
00674  63 00000 10000
00675  55 10000 00006
00676  63 00000 10000
00677  11 01450 20000
00700  47 00701 00702
00701  63 00000 00336
```

DICTIONARY OF REGIONAL ADDRESSES AND EQUIVALENTS AT 01470 THRU 01633, AND 01634 THRU 01777

TRANSLATE AND PUNCH INSERT ADDRESS ROUTINE. ADDRESSES - 00660 THRU 0721

PX 71900-9-(126)

```
00702  63 00000 00216
00703  11 00270 01465
00704  11 00357 01466
00705  11 00241 01435
00706  11 00241 01436
00707  11 01434 01440
00710  63 00000 00267
00711  75 00006 00713
00712  63 00000 00255
00713  63 00000 00267
00714  63 00000 00255      TRANSLATE AND
00715  11 00270 01433        PUNCH CHECK
00716  55 01440 00025      ADDRESS, AND
00717  55 00250 10011        CHECK SUM ROUTINE.
00720  37 00565 00544        ADDRESSES-
00721  45 00000 00403        00722 THRU 00757
00722  37 00472 00465
00723  43 01434 00725
00724  45 00000 01270
00725  45 00000 00726
00726  75 00007 00730
00727  63 00000 00255
00730  11 20000 01440
00731  55 01440 00030
00732  51 00253 20000
00733  35 00254 00734
00734  63 00000 00000
```

PX 71900-9-(126)

```
00735  63 00000 00267
00736  11 00271 01433
00737  55 00250 10012
00740  37 00565 00544
00741  45 00000 00742
00742  11 00272 01440
00743  37 00721 00711
00744  11 01435 10000
00745  37 00565 00540
00746  11 01436 10000
00747  37 00565 00540
00750  11 00273 20000
00751  37 00741 00726
00752  16 00274 00721
00753  16 00275 00741
00754  63 00000 00337
00755  45 00000 00403
00756  21 01464 00246    TRANSLATE AND
00757  45 00000 00722    PUNCH PROGRAM START
00760  21 01464 00246    ROUTINE.
00761  37 00472 00465    ADDRESSES-
00762  45 00000 00763    00760 THRU 01003
00763  11 00241 01447
00764  35 00277 01442
00765  11 00301 01434
00766  37 00721 00667
00767  11 01442 10000
00770  37 00565 00530
00771  11 00302 20000
```

```
00772  37 00755 00726
00773  11 00241 01434
00774  37 00721 00667
00775  11 01442 10000
00776  37 00565 00530
00777  11 00246 20000
01000  37 00755 00726
01001  16 00274 00721
01002  16 00274 00755
01003  45 00000 00403
01004  21 01464 00246
01005  37 00604 00600
01006  37 01166 01156
01007  11 01440 01443
01010  37 01206 01203
01011  11 20000 01444
01012  37 00577 00571
01013  43 00267 01015
01014  45 00000 01276
01015  75 10012 01017
01016  11 00241 01445
01017  16 00332 01024
01020  37 00577 00571
01021  43 00215 01027
01022  43 00217 01027
01023  37 01206 01203
```

DECIMAL
NUMBER
INPUT
CONVERSION ROUTINE
ADDRESSES -
01004 THRU 01244

```
01024  11 20000 01445
01025  21 01024 00246
01026  45 00000 01020
01027  11 00307 10000
01030  51 01024 01440
01031  11 00333 20000
01032  42 01440 01277
01033  37 00604 00600
01034  37 01202 01167
01035  11 01440 01457
01036  11 01430 20000
01037  73 00327 01460
01040  11 20000 01461
01041  37 00604 00600
01042  45 00334 01051
01043  37 01202 01167
01044  11 01440 20000
01045  47 01046 01047
01046  13 01430 01430
01047  11 01430 01462
01050  11 00341 20000
01051  11 20000 01432
01052  37 01133 01056
01053  11 01430 10000
01054  37 00565 00530
01055  45 00000 00403
```

```
01056. 31 01456 00000
01057  75 30012 01061
01060  72 01444 00316
01061  73 00316 01430
01062  54 20000 00043
01063  73 00316 01431
01064  31 01430 00043
01065  32 01431 00000
01066  75 10013 01070
01067  11 00241 01444
01070  74 20000 01444
01071  46 01074 01072
01072  31 20000 00000
01073  45 00000 01075
01074  21 20000 00246
01075  74 20000 01445
01076  11 20000 01430
01077  23 01444 00337
01100  35 01445 01444
01101  37 01244 01207
01102  71 01430 01446
01103  74 20000 01450
01104  46 01107 01105
01105  31 20000 00000
```

```
01106  45 00000 01110
01107  21 20000 00246
01110  74 20000 01451
01111  11 20000 01430
01112  21 01444 01447
01113  35 01450 20000
01114  35 01451 01444
01115  11 01432 20000
01116  47 01134 01117
01117  21 01444 01462
01120  47 01121 01124
01121  46 01122 01302
01122  42 00341 01300
01123  35 00340 01444
01124  16 01444 01125
01125  54 01430 00000
01126  46 01127 01130
01127  35 00246 01430
01130  11 01443 20000
01131  47 01132 01133
01132  13 01430 01430
01133  45 00000 00000
01134  21 01444 00337
01135  35 00345 01444
```

```
01136  21 01430 00345
01137  43 20000 01142
01140  21 01444 00246
01141  31 00345 00033
01142  11 20000 01430
01143  31 01443 00000
01144  47 01145 01146
01145  13 01430 01430
01146  11 01444 20000
01147  46 01300 01150
01150  42 00346 01153
01151  43 00346 01153
01152  45 00000 01302
01153  11 00346 10000
01154  53 01444 01430
01155  45 00000 01133
01156  43 00335 01162
01157  43 00336 01164
01160  11 00241 01440
01161  45 00000 01166
01162  11 00246 01440
01163  45 00000 01165
01164  11 00241 01440
01165  37 00577 00571
01166  45 00000 00000
```

```
01167  37 01166 01156
01170  37 01206 01203
01171  11 20000 01430
01172  37 00577 00571
01173  43 00215 01202
01174  43 00216 01202
01175  43 00217 01202
01176  37 01206 01203
01177  72 01430 00387
01200  11 20000 01430
01201  45 00000 01172
01202  45 00000 00000
01203  75 20012 01301
01204  43 00220 01205
01205  51 00240 20000
01206  45 00000 00000
01207  11 00342 01446
01210  11 00343 01447
01211  41 01460 01213
01212  45 00000 01222
01213  71 01446 00316
01214  74 20000 01450
01215  46 01216 01217
01216  21 20000 00246
01217  11 20000 01446
01220  21 01447 01450
01221  45 00000 01211
01222  11 00331 20000
```

```
01223  36 01461 01224
01224  71 01446 00330
01225  74 20000 01450
01226  46 01227 01230
01227  21 20000 00246
01230  11 20000 01446
01231  21 01447 01450
01232  11 01457 20000
01233  47 01234 01244
01234  31 00246 00105
01235  73 01446 01456
01236  31 20000 00001
01237  42 01446 01241
01240  21 01456 00246
01241  13 00344 20000
01242  36 01447 01447
01243  11 01456 01446
01244  45 00000 00000
01245  21 01464 00246
01246  45 00000 01247
01247  37 01264 01255
01250  56 00000 00360
01251  21 01464 00246
01252  45 00000 01253
```

END . (OR END)
OF ASSEMBLY
ROUTINE.
ADDRESSES -
01245 THRU 01265

```
01253  37 01264 01264
01254  45 00000 70011
01255  75 00074 01257
01256  63 00000 00241
01257  63 00000 00267
01260  63 00000 00255
01261  63 00000 00255
01262  63 00000 00267
01263  63 10000 00337
01264  75 00144 30000
01265  63 00000 00241
01266  45 00000 01323
01267  45 00000 01331
01270  45 00000 01345
01271  45 00000 01333
01272  45 00000 01357
01273  45 00000 01335
01274  03 16121 61630
01275  12 30300 30322
01276  45 00000 01337
01277  45 00000 01341
01300  45 00000 01353
01301  45 00000 01343
01302  45 00000 01355
01303  00 00000 00000
01304  26 20260 33406
01305  22 22031 70000
```

ERROR DETECTION

OUTPUT (TYPEWRITER)

ROUTINE.

ADDRESSES

01266 thru 01427

```
01306  37 00577 00571
01307  43 00216 01311
01310  45 00000 01306
01311  21 01434 00246
01312  41 00347 01314
01313  45 00000 00403
01314  16 00314 00531
01315  16 00274 00666
01316  16 00274 00725
01317  16 00274 00762
01320  16 00315 01246
01321  16 00350 01252
01322  45 00000 00403
01323  55 01274 10006
01324  51 00212 01441
01325  55 10000 00006
01326  51 00212 01442
01327  37 01403 01367
01330  45 00000 01306
01331  55 01274 10022
01332  45 00000 01324
01333  55 01275 10006
01334  45 00000 01324
01335  55 01275 10036
01336  45 00000 01324
01337  55 01304 10006
01340  45 00000 01324
```

```
01341  55 01304 10022
01342  45 00000 01324
01343  55 01305 10006
01344  45 00000 01324
01345  55 01274 10036
01346  51 00212 01441
01347  55 10000 00006
01350  51 00212 01442
01351  37 01403 01367
01352  45 00000 01311
01353  55 01304 10036
01354  45 00000 01346
01355  55 01305 10022
01356  45 00000 01346
01357  55 01275 10022
01360  51 00212 01441
01361  55 10000 00006
01362  51 00212 01442
01363  37 01403 01367
01364  11 00472 20000
01365  75 20005 01306
01366  43 00352 01311
01367  61 00000 00216
01370  61 00000 00213
01371  61 00000 01441
01372  61 00000 01442
```

```
01373  61 00000 00215
01374  61 00000 00214
01375  11 01463 20000
01376  37 01420 01404
01377  61 00000 00215
01400  11 01464 20000
01401  37 01420 01404
01402  61 00000 00216
01403  45 00000 00000
01404  75 30004 01406
01405  73 00325 01446
01406  15 00331 01410
01407  11 00304 01454
01410  11 01446 20000
01411  47 01421 01412
01412  61 00000 00215
01413  21 01410 00306
01414  41 01454 01410
01415  11 00266 20000
01416  36 01451 01417
01417  61 00000 00000
01420  45 00000 00000
01421  15 01410 01423
01422  11 00266 20000
01423  36 01446 01424
01424  61 00000 00000
01425  21 01423 00306
01426  41 01454 01422
01427  45 00000 01415
```

Library Routine

Introduction

The ERA Library Routine for the 1103 Serial 9 Computer extracts subroutines from a library file on magnetic tape and assembles them on the drum or in high speed storage for later use. At the option of the operator, a flex tape of all subroutines assembled is punched out on the high speed punch.

This routine may be used in conjunction with RECO II (ERA Regional Coding Routine II) to give a continuous flex tape consisting of a main routine followed by all subroutines needed in the execution of the main routine. A description of this overall assembly procedure is given in the write-up of RECO. It is also possible to use the Library Routine separately. In either case input information must be provided by the operator in the form of a flex tape. This tape lists the subroutines to be assembled, together with information regarding assembly modification and storage. Routines may be stored at a location different from that with respect to which they are assembly modified. A detailed description of the input tape is given in the section "Input Tape Format".

Main Features of the Library Routine

1) Exit Restoration

The Library Routine replaces the first two words of subroutines in "standard form" with the instructions

37 75700 75702

45 00000 [30000]

The first instruction refers to the Convair Alarm Routine which has had its insert address changed from 76000to 75700 in order to avoid interfering with the image. The second instruction is the standard exit. This "exit restoration" feature is included to facillitate relocating the Alarm Print routine. In fact, with minor modifications, a different standard preface can be appended to subroutines in the tape library, should this ever be desired.

2) Assembly Modification

Any routine coded may be modified relative to any HSS or drum address subject to the following restrictions:

a) The routine is coded relative to 01000.

b) All instructions and relative operands are located consecutively at the beginning of the routine.

c) All absolute constants are stored consecutively immediately after the modifiable words.

A listing of routines which may be modified by the library routine is given in the appendix. The programmer must choose for each modifiable routine, the starting address with respect to which address modification is to occur, as explained in detail in the section on "Input Tape Format".

3) Storage of Subroutines

The programmer may choose to store a subroutine beginning at any HSS or drum address except in the image (76000-77777). Storage in the image will result in simultaneous storage of the subroutine in the corresponding high speed storage location.

It is the programmer's responsibility to be sure that stored routines do not overlap. If it is desired to store two or more routines in the same locations, then the library routine must be applied two or more times.

The programmer must choose a storage location for each routine whether it is assembly modifiable or not.

The method of providing information on desired storage locations is given in the section "Input Tape Format".

4) Typed List of Subroutines

During the operation of the routine, a list of subroutines together with other information is typed out. This list may be used by the programmer to verify that the subroutines have in fact been assembled as he intended. For details see the section "Typed List of Subroutines".

5) Flex Tape Output of Assembled Routines

The operator may procure flex tapes of the assembled subroutines if he desires (see Operating Instructions). In general either all of the subroutines are punched out, or none of them are. The following flex tape is punched out for each subroutine (with appropriate leader or trailer):

a) Insert Address (First Storage Address).

b) Subroutine (modified or not as the case may be).

c) Check Address.

d) Insert address 75202.

e) Check sum.

f) Check Address 75204.

6) Final State after Using Routine

After the routine has been used and all subroutines have been assembled (and possibly punched out), HSS is restored to its original form, except for those locations in which subroutines have been stored. All the subroutines are in their desired locations ready for use. Magnetic Tape Unit 0 has been backed up to its starting point.

The Subroutine Library

A list of subroutines currently stored on MT Unit 0 is given in the appendix. Preceding the first word of each subroutine on magnetic tape are four "tag words". Following the last word of each subroutine are enough "all zero" words to fill out the block of tape on which the last word appears. For example, a subroutine containing 46_8 words occupies two blocks on the tape; i.e., 4 tag words, 46_8 subroutine words, and 26_8 blanks.

The four "tag words" preceding a subroutine give the following information:

1). First word: b c ddddd eeeee

b = 0 if routine is assembly modifiable.

b = 1 if routine is not assembly modifiable.

c = 0 if exits are not to be restored.

c = 1 if exits are to be restored.

ddddd: number of words if routine is assembly modifiable; first fixed operating location if it is not.

eeeee: number of modifiable words if routine is assembly modifiable; last operating location if it is not.

2). Second Word: Check Sum (Left Part).

3). Third Word: Check Sum (Right Part).

4). Fourth Word:

This word is the routine's library index, or more simply, Routine Index, and consists of a flex code of five letters, which are determined by the nature of the routine and two octal zeros to make up a 36 bit word. Examples:

a). Central Exchange Index: RR-68 (e^x - Floating Point).

Routine Index: ex - f01

Routine Identification Code: 20 27 26 37 53 00

b). Central Exchange Index: CV 37 (Card Package)

Routine Index: cg - 100

Routine Indentification Code: 16 13 52 37 37 00

(Note that the "dash" in the index does not appear in the identification code.)

The Library Routine is also stored on MT Unit 0. It is transferred to HSS from tape by the Library Routine loader in the Service Library.

The Dictionary

Immediately following the Library Routine on MT Unit 0 is a 40 block "dictionary". The first half consists of a 2 word check sum plus up to 510 (decimal) identification codes of the type described in the preceding section. All words not filled with identification codes are filled with zeros.

The second half of the dictionary consists of a 2 word check sum and up to 510 "control words". For each identification code in the first half of the dictionary, there is a control word in the same relative position in the second half; i.e., if the identification code corresponding to a given subroutine is the nth word in the first half of the dictionary, $3 \leq n \leq 512$, then the "control word" for this subroutine is also in position n, but in the second half of the dictionary. The control word consists of 12 octal digits

op ppp qq 00000 , where

pppp is the number of blocks of tape in MT Unit 0 measured from its start which must be traversed to get to the start of the subroutine,

$0364_8 \leq pppp \leq 2047_8$;

qq is the number of blocks in the subroutine, $01 \leq qq \leq 20_8$.

If a subroutine is more than 20_8 blocks in length it is regarded as two or more separate routines, each with its own "tag words" and "index".

As an example, the control word 01 24603 00000 calls for a routine which starts after 1246 blocks of tape have been counted and which is 3 blocks long. The subroutine therefore occupies blocks 1247, 1250, and 1251. (octal)

Input Tape Format

The input tape lists the subroutines which are to be used in a main routine together with information regarding assembly modification and desired storage locations. The programmer must provide three pieces of information for each subroutine to be assembled.

The exact format of this information depends on whether the routine is assembly modifiable.

1.) For an assembly modifiable routine:
   a) Routine Index
   b) An assembly modification address; i.e. the first operating location of the subroutine.
   c) A storage address; i.e. the address where the first word of the subroutine is to be stored.

2.) For a subroutine which is not assembly modifiable, (i.e. the subroutine is written with a fixed operating location or is written relative to H.S.S. location 01000 but is not in "standard form").
   a) Routine Index
   b) Fixed operating address (i.e. the first address of the routine.)
   c) A storage address

A maximum of 16 subroutines can be assembled with one application of the Library Routine. Care must be taken not to store two routines in the same location, since all subroutines are assembled and stored before

any of them are punched. If the operator desires to store more than one subroutine at the same location, he must use the library routine more than once and must have a "separate" routine list for each application of this routine. (The tape need not be separated literally, but must have the proper "terminating signal" between each routine list.)

At the conclusion of each routine list is a period. This tells the machine that all input information has been received and transfers operation to the next part of the routine. If more than 16 subroutines are called for without a period, the input tape is automatically stopped after reading in 16 indices, and the computer halts after assembling the 16 subroutines. Push start button to assemble the remaining subroutines.

The exact format of the tape is best illustrated by an example:

Six subroutines are to be assembled. The subroutine library representation and Central Exchange Numbers of these subroutines are:

1) rt-f01 (RR 59)
2) cp-100 (CV 1)
3) do-x01 (RR 47)
4) al-pr1 (CV 3)
5) ip-p01 (RR 10)
6) rt-x01 (RR-21)

In the above list, routines 1, 3, and 6 are written relative to H.S.S. address 01000. Routine 5 is so written on tape as to be assembly modifiable. Routines 2 and 4 operate from fixed locations.

These routines are to be operated on as follows:

1) rt-f01 modified relative to 1300 and stored beginning at 51000.
2) cp-100 has fixed operating location 00040 but is to be stored at 42100.

3) do-x01 is not to be modified but is to be stored at 66300.

4) al-prl has fixed operating location 75700 and is to be stored there.

5) ip-p01 is not to be modified but is stored at 41000.

6) rt-x01 is to be modified relative to 01272 but is to be stored at 00300.

The information needed on the input tape consists of the following in flex code:

rt-f01	01300	51000
cp-100	00040	42100
do-x01	01000	66300
al-prl	75700	75700
ip-p01	01000	41000
rt-x01	01272	00300

The following general rules are noted:

1) Three pieces of information are needed for each subroutine.

2) If a "standard" assembly modifiable routine is not to be modified, the address 01000 is the first address listed; otherwise the starting address is given.

3) The first address for a routine which has a fixed operating location is the fixed operating location.

4) The first address for a routine written relative to 01000 but which is not in "standard form" (and hence is not modifiable) is 01000.

5) The second address is always the first address at which the subroutine is to be stored.

6) An assembly modifiable routine may be modified relative to any H.S.S. or drum address.

3) A routine may be stored anywhere in H.S.S. or on the drum. Care must be taken not to store more than one subroutine in the same location.

In preparing the input flex tape, the following rules should be observed:

1) There must be at least one space between the two letters and the three (or more) digits of the library index.
2) There must be at least one space between the library index and the first address and between the first and second addresses.
3) There must be a carriage return after the second address.
4) Following the carriage return at the end of the last subroutine information there must be a period.
5) The library routine ignores extra spaces, code deletes and leader on the input tape.

Typed Lists of Subroutines

As the library routine is operating on subroutines taken off magnetic tape, it types out information for later reference.

1) If the subroutine is modifiable and has been modified, the following is typed:
   a) IL03 Library Index
   b) Number of Words Number of Modifiable Words
   c) MOD aaaaa
   d) bbbbb ccccc

where: aaaaa is the starting address
bbbbb is the first storage address
ccccc is the last storage address

2) If the subroutine is modifiable but has not been modified, the same information is typed as in 1 (except that c) is "NOT MODIFIED".

3) If the subroutine has a fixed operating location the following is typed:

a) 1103 Library Index

b) ddddd eeeee

c) bbbbb ccccc

where: ddddd is the first operating location

eeeee is the last operating location

bbbbb and ccccc are as in 1

In the example previously discussed the following would be typed (although not necessarily in this order).

```
rt-f01
00056    00050
mod      01300
51000    51055
op-100
00040    00077
42100    42137
do-x01
00202    00160
not modified
66300    66501
al prl
75700    75643
75700    75643
```

ip-p01

00422 00421

not modified

01000 01421

rt-x01

00047 00040

mod 01272

00300 00350

The programmer may quickly scan this list to see if the proper subroutines were assembled and if there is any overlap in storage.

Error Detection - Alarms

The library routine detects certain errors during the course of its operation. These errors are indicated by typewritten symbols. After encountering one of the errors described below, the machine stops. In most cases, several options are available to the operator, as outlined below.

1. "a2" - check sum failure on transfer of the library routine to HSS. MT Unit 0 is backed to the beginning of the Library Routine Press Start to retransfer routine.
2. "c1" - check sum failure on transfer of the first half of the dictionary to H.S.S. The magnetic tape is backed to the beginning of the dictionary.
   a) To retransfer first half of dictionary: Press START
   b) To ignore check sum failure: set PAK to 00600 and START
3. "R" - a non-existent index has been read into the machine. Press START. The non-existent index is ignored, and the program continues.
4. "C2" - check sum failure on transfer of second half of dictionary

into H.S.S. The magnetic tape is backed to the beginning of the second half of the dictionary.

a) To retransfer second half of dictionary: Press START

b) To ignore check sum failure: Set PAK to 00602 and START

5. "C3" - check sum failure on transfer of subroutine from library to H.S.S. The magnetic tape is backed to the start of the subroutine.

a) To retransfer subroutine: Press START

b) To transfer next subroutine (or to proceed with program if failure was on last subroutine):

Set PAK to 00604 and START

c) To ignore check sum failure: Set PAK to 00606 and START

6. "C4" - The address portion of the input flex tape is incorrect (e.g. less than five octal digits, a character is used other than 0, . . ., 7). Press START to bring in next index (or continue program).

If an error is indicated (other than "C1"), one of the subroutines does not get assembled. A look at the typed list of subroutines determines which subroutine is missing.

Operating Instructions

1) Prepare a flex tape which calls for the subroutines wanted and which provides information on assembly modification and storage. This tape may be spliced onto the end of input tape used for RECO II when both routines are to be used in succession (See discussion of this in the RECO II write-up). If this routine is to be used alone, only the subroutine flex tape is needed. (For information on this input flex tape see the section "Input Tape Format").

SUBROUTINE LIBRARY APPENDIX

Subroutines Currently on Magnetic Tape 0

		Central Exchange Index	Library Code
a.	Constant Pool	CV-1	cp-l00
b.	Alarm Print	CV-3	al-prl
c.	Card Package	CV-37	cg-l00
d.	Interpretive System, unpacked	RR-10	ip-p01
e.	Interpretive System, packed	RR-10	ip-p01
f.	Decimal Output Floating Point	RR-20	do-f01
g.	Decimal Output Fixed Point	RR-47	do-x01
h.	Square Root Floating Point	RR-59	rt-f01
i.	Square Root Fixed Point	RR-21	rt-x01
j.	e^x, Fixed Point	None	ex-x01
k.	e^x, Floating Point	RR-68	ex-f01
l.	Arcsin/cos x, Floating Point	RR-75	tg-f20
m.	Arctan x, Floating Point	RR-74	tg-f40
n.	Sin $\frac{1}{2}$, cos $\frac{1}{2}$ x, Fixed Point	RR-24	tg-x10
o.	Sin x, cos x, Fixed Point	None	tg-x11
p.	Arctan x, Fixed Point	RR-26	tg-x40
q.	Sin $\frac{1}{2}$, cos $\frac{1}{2}$ x, Floating Point	RR-62	tg-f10

As more subroutines are put in the library, supplements to this list will be issued from time to time.

bpj

2) Turn ON PUNCH and TYPEWRITER

3) BE SURE MT UNIT 0 is ON

4) Set Jump Switch #1 if a flex tape of the subroutines is NOT desired.

5) If RECO II is not used, set PAK to 70011 and START. If RECO II is used, this routine will start automatically after completion of RECO II.

6) If an alarm "C1", "C2", "C3", "C4", or "R" is typed, the machine stops. For a choice of procedures, see the discussion of "ERROR DETECTION - ALARMS". In any case pressing the START will cause the Library Routine to continue; however, a subroutine may be omitted.

7) The routine halts at 56 00000 70011

8) To repeat the library routine, press START after the 56 stop.

NOTE: If a period is inadvertently omitted at the conclusion of of a Routine List, Force Stop Computer after Input Tape has completely passed through reader set PAK to0052 and START.

REMINGTON RAND UNIVAC

Information Science Division

A Multiple Regression and Correlation Program for the Univac Scientific (1103)

1 July 1956

WRITTEN AND MACHINE CHECKED BY:

D. C. McGowan - California Research Corp.

Jack Rose - Remington Rand Univac

Leo Kennedy - Remington Rand Univac

THE 1103 MULTIPLE REGRESSION AND CORRELATION PROGRAM

General Description

There is now available to Univac Scientific (ERA 1103) users and service bureau customers a library program for multiple correlation (regression) analysis. The program will handle up to m=30 variables (including the dependent) and up to n=400 observations of each. The output, on punched cards, is:

1. The identification number of each variable used.
2. The mean of each variable used.
3. The standard deviation of each variable used. (Based on n-1 degrees of freedom.)
4. The normalized regression coefficients (regression coefficients for each variable reduced to standard units).
5. The ordinary regression coefficients.
6. The simple correlation coefficients.
7. The partial correlation coefficients involving the dependent variable. (The remaining partial correlation coefficients may be obtained if desired.)
8. The inverse of the correlation matrix (optional).
9. The square of the multiple correlation coefficient, the standard error of estimate (based on n-m degrees of freedom), and the constant term in the regression equation.

PX 71900-9-(127)

10. (Optional) The back solution, in which the regression equation just computed is used to predict a value of the dependent variable for each observation. The output consists of a card for each observation containing:
    a. The number of the observation and the identification number of the dependent variable.
    b. The predicted value of the dependent variable.
    c. The observed value of the dependent variable.
    d. The difference (b-c).

Optional items may be obtained by making manual jump selections on the computer console.

There is also available a program for the back solution (10 above) in the event that the regression equation is already known and no correlation analysis is desired. In this program, the output is as follows, with the numbers having the same significance as above:

1. Identification of variables used.
5. As before. (Reprint of input to avoid error or ambiguity.)
9. As before (same).
10. As before.

The programs are designed to make good use of the great facility of the 1103 for this type of computation. Also,

they offer a number of special features, in addition to those listed above, that make them very flexible and economical. For these reasons they are extremely fast and will give more information at less cost than any other library program up to the maximum size of problem they will handle.

The Nature of the Problem

It frequently happens that a dependent variable, y or x_1, can be considered to be a linear function of a number of independent variables, $x_2, x_3, \ldots x_m$. Then:

$$(1) \qquad y^* = x_1^* = a_0 + \sum_{j=2}^{m} a_j x_j$$

where $y^* = x_1^*$ is an estimate of y (or x_1).
This implies that the effect of each x on y is linear and independent of the levels of the other variables. However, when this is not the case, it is often possible to make a more realistic mathematical model of the situation and with little or no distortion to apply a transformation that will linearize it to the above form. When this is done, the x's will not generally be the directly measured experimental quantities but functions of them, or functions of groups of them, thus accounting for nonlinearity and interaction effects.

The multiple correlation problem proper begins when the functions for the x's have been chosen so that an equation in the form of (1) has been arrived at and when there are available n observations, n>m, in each of which values of all variables have been measured. The problem is then to choose a set of values for the coefficients, "a," that will best fit all the data points.

This problem is solved with the most computational ease by classical least-squares fitting, which defines a_0 as

$$(2) \qquad a_0 = \bar{x}_1 - \sum_{j=2}^{m} a_j \bar{x}_j$$

and determines values for the regression coefficients a_2, a_3, . . . a_m. Certain other statistics are also computed which indicate how well in general the model has succeeded in representing the effects of the variables and predicting the values of x_1, and to what extent the effect of each variable is numerically important, statistically significant, and successfully represented by the model chosen, Equation (1).

Notation

Let $y = x_1$ be the dependent variable. Let x_2, x_3, . . . x_m be the independent variables. Let the index i or j designate the number of the variable:

$$i = 1\ (1) \ldots m$$

$$j = 2\ (1) \ldots m$$

where m is the total number of variables, including the dependent variable.

In general, the term "the x_j's" will refer to the variables x_2 to x_m, but not to x_1 or y.

Let the index h refer to the run, or observation, number:

$$h = 1\ (1) \ldots n$$

where n is the total number of observations of each variable. (For the case of missing observations, see the section "Special Features.") Thus x_{hi} is the value of the i'th variable observed in run number h.

Other conventions of notation will be defined as they are used.

The Method of Computation

In the 1103 program, the mean of each variable, $\bar{x}_i$, is first computed:

$$(3) \qquad \bar{x}_i = \frac{\sum_{h=1}^{n} x_{hi}}{n}$$

The standard deviation, based on n-1 degrees of freedom, is then computed for each variable by:

$$(4) \qquad s_i = \frac{\sqrt{\Sigma_h x_{hi}^2 - \bar{x}_i \Sigma_h x_{hi}}}{\sqrt{n-1}}$$

where Σ_h designates summation over all runs with the square roots being taken separately by the computer, and the other arithmetic operations performed as indicated here.

The simple correlation coefficients between all pairs of variables are then computed by:

$$(5) \qquad r_{ij} = \frac{\Sigma_h x_{hi} x_{hj} - \bar{x}_i \Sigma_h x_{hj}}{\sqrt{\Sigma_h x_{hi}^2 - \bar{x}_i \Sigma_h x_{hi}} \sqrt{\Sigma x_{hj}^2 - \bar{x}_j \Sigma_h x_{hj}}}$$

The above computations are all computed in fixed binary with precision to full single-word capacity being retained. The simple correlation coefficients are stored on the drum for printing out and then converted to normalized floating binary, mantissa 27 bits plus sign bit.

The matrix of simple correlation coefficients, $R = (r_{ij})$ is then inverted, in floating binary. In the array

of this matrix, the correlation coefficients involving the dependent variable, x_1, that is, the r_{1j} coefficients, are in Row 1 and Column 1. The inversion is accomplished by a direct elimination without any permutation of rows and columns. R is symmetrical, and the inverse is forced to be symmetrical, regardless of round-off error, by computing only the diagonal and one triangle and assuming that the other triangle will be the mirror image. The inverse so computed is Q_1. $Q_k \cong R^{-1}$, where R^{-1} is identically equal to the inverse of R and Q_k is the k-th approximation of R^{-1}. Q_1, the first approximation, is then reconverted to fixed binary.

Unless R is badly conditioned, round-off error will be small, but the inverse is in any event improved by the formula

$$(6) \qquad Q_{k+1} = Q_k(2 - RQ_k)$$

somewhat modified. Here 2 is the diagonal 2 matrix. The modification of the above formula consists of the fact that Q_{k+1} is also forced to be symmetrical as was Q_1. Although Q_k and R are both symmetrical, RQ_k, and consequently $(2 - RQ_k)$, are not necessarily symmetrical; and, indeed, the full square matrix is computed. When this square matrix is multiplied by Q_k, however, only one triangle of the product matrix, Q_{k+1}, is computed.

In this improvement scheme, the square matrix $(2-RQ_k)$, which will be approximately the unit matrix, is computed in the first half of each iteration cycle. The sum of the absolute values of 1 minus each diagonal element of this matrix is then

compared with the corresponding sum obtained for the matrix $(2-RQ_{k-1})$. (On the first cycle, it is compared to a very large arbitrary number.) If this value is larger than for the previous iteration, the improvement method is assumed not to be converging on the true inverse. If it is smaller, the second half of the iteration cycle is performed by multiplying the square matrix $(2-RQ_k)$ by the matrix Q_k to form the (arbitrarily) symmetrical matrix Q_{k+1}. The same value,

$$\sum_{i=1}^{m} \left| \left[1 - (rq_k)_{ii} \right] \right|$$

is then compared with a "tolerance" number to see if Q_k was a sufficiently close approximation to R^{-1}. This tolerance number is established by multiplying an arbitrary small number by the number of diagonal elements in RQ; when the sum of absolute values by the above formula is smaller than this product, $(q_k)_{ij}$ will, on the average, equal r^{-1}_{ij} to five or six correct decimal digits to the right of the decimal point. At this time, however, Q_{k+1}, a still better approximation of R^{-1}, has already been computed and will be used as the final approximation.

If, on the other hand, the above test shows that the sum of absolute values of 1 minus each diagonal element of $(2-RQ_k)$ increases from one cycle to the next, the original inverse, Q_1, is then taken as the best approximation of R^{-1}; and the problem can be finished using this, the typewriter indicating that this has occurred. Space is left in the program for a section of code (to be written if experience proves it desirable) that will apply a more powerful improvement scheme

on failure of the above scheme to converge. The alternate scheme would be $Q_{k+1} = Q_k(RQ_k)^{-1}$, which is slower but which will always converge, unless R or Q_1 is singular.

The improved inverse (or the unimproved inverse, if the improvement scheme has failed to converge) is then used to compute the square of the multiple correlation coefficient, the standard error of estimate, the normalized regression coefficients, the regular regression coefficients, and the partial correlation coefficients.

The square of the multiple correlation coefficient is computed by:

$$(7) \qquad (r_{1.2,3,\ldots m})^2 = 1 - 1/q_{11}$$

(q_{ij} is the best approximation of r_{ij}^{-1})

The standard error of estimate, s_e, based on n-m degrees of freedom, is then computed by:

$$(8) \qquad s_e = \frac{s_1}{\sqrt{q_{11}}} \sqrt{\frac{n-1}{n-m}}$$

since s_1 was based on n-1 degrees of freedom.

The normalized regression coefficients (regression coefficients expressed in standard units) are computed by:

$$(9) \qquad b_j = -q_{1j}/q_{11}$$

The regular regression coefficients (the a_j's of Equation 1) are computed by:

$$(10) \qquad a_j = b_j(s_1/s_j)$$

The partial correlation coefficients, p_{ij}, are computed by:

$$(11) \qquad p_{ij} = \frac{-q_{ij}}{\sqrt{q_{ii}}\,\sqrt{q_{jj}}}$$

Finally, the constant term in the regression equation, the a_0 of Equation 1, is computed by applying Equation 2. This completes the regression analysis proper, but if a back solution is desired, Equation 1 is then applied to each of the n observations to compute an estimated value of y for each run.

Special Features

The 1103 multiple correlation routine has a number of special features designed to make the program unusually useful, flexible, and economical of machine time. Some, such as listing the normalized regression coefficients and the use of a matrix inversion routine tailored specifically for this problem, are fixed parts of the code, but others are optional and under the control of the operator. The latter are listed here.

1. Independent Variables on Magnetic Tape

In some types of problems, a single set of independent variables may be used with a number of different dependent variables. In this case, the use of each new dependent variable constitutes an entire new problem, but important amounts of read-in time can be saved by writing the independent variables, the x_j's, on magnetic tape on the first problem and reading them from this tape in subsequent problems. The program is so designed that selecting MJ No. 1 will write the x_j's on Tape Unit 2 as they are read in from cards. Also, it is possible in any run to read the x_j's either from cards or from Tape Unit 2. The y's are always read in from cards and are never written on tape. When the x_j's are written on tape, they are summed, and the sum is written as the last block of data on the tape. When they are read from tape, they are summed again, and the sum is compared with that written on the tape; if there is a discrepancy, the on-line typewriter types out "Tape sum no good," or a suitable abbreviation thereof. Unless otherwise indicated by the

typewriter, the x_j's can be assumed to be correctly read in from MT 2.

2. Elimination of Undesired Variables

The results of the correlation analysis may show that the effects of certain independent variables on the dependent variable are statistically insignificant or numerically unimportant. It may then be desired to rerun the correlation excluding these variables. This is easily done in the program whether the x_j's are read in from cards or from tape.

When all data are to be read in from cards, a control card is used bearing the numbers m and n. Omission of an undesired variable can be accomplished simply by omitting the corresponding deck of data cards and using a new control card with the altered value of m.

When the X_j's are to be read in from tape and only x_1 from cards, the control deck consists of a card bearing the number of variables to be omitted, followed by as many cards as there are variables to be omitted, each bearing the code number of one x_j to omit. When no variables are to be omitted, the control deck consists of one blank card. In reading from tape, the machine reads all the observations of one x_j, sums them, and then scans the list of variables to be omitted to see whether to reject this x_j. In any case, the sum is retained, so that the sum check on the tape reading is preserved.

3. Treatment of Missing Observations

No missing observations among the independent variables are permitted. If a run does not contain a full set of

observations, it should either be rejected or reasonable values for the missing observations should be estimated. However, when a single set of independent variables is used with a number of different dependent variables, it often happens that the sets of observations of some of the dependent variables are not complete. In this case, the runs with missing observations are rejected, but it is convenient to let the machine do this rejecting so that it can always be fed the same set of data for the independent variables. This is done by representing each missing observation on the deck of y's by -0. The machine will scan the observations of y (but not of the x_j's) looking for -0. (Plus 0 is a legitimate observation whose value happens to be zero and will be treated normally.) When it finds a -0 for a y_h it changes it to normal zero, it sets all the corresponding observations x_{hj} equal to zero, and it subtracts one from the value of n. The effect in the main part of the problem is as if that run had never been included in the data.

When the back solution is run with the data still in the machine, y_h^*, the predicted value of the dependent variable for a missing observation, is a_0, since y* is computed by Equation 1 and all the x_{hj}'s have been set to zero. The observed value is listed out as +0.

When the back solution is run later, as a separate problem, the tally program that rejects missing observations does not operate. In this case, y_h^* assumes the value dictated by Equation 1 for the set of values x_{hj} and for the previously

computed a_j's. The observed value of y_h is listed as it was punched, -0.

4. Typing of the Square of the Multiple Correlation Coefficient

It is sometimes convenient to test different functional representations of the dependent variable ($y=v$, $y=\log v$, $y=e^v$, etc., where v is the quantity directly measured) against the same set of independent variables. In this case, the choice of functions for best fit can often be made solely on the value of the multiple correlation coefficient. The square of the multiple correlation coefficient is therefore typed out as soon as it is computed so that the machine can be stopped and the problem abandoned at this point if desired. This number is also listed on cards with the normal output of the program.

5. Suppression of Most of the Partial Correlation Coefficients

In many problems, the only partial correlation coefficients of interest are those between the dependent variable and the independent variables, that is, p_{1j}. The partial correlation coefficients of the independent variables with each other (p_{ij}, $i \neq 1$) are usually of interest only when a new mathematical model for the regression relationship is being tested. Therefore, to save time, the program normally computes only the partials p_{1j}. However, if the remaining partials are desired, they will be computed and listed out if manual jump selection No. 2 is made. The time required will be about one second for each ten additional partials.

6. Suppression of Listing the Inverse Matrix

In many routine problems, the inverse of the correlation matrix is not of particular interest, and considerable time is saved by not listing it out. Therefore, the program will compute but not normally list out the inverse. However, when it is desired to see the inverse, it will be listed out by selecting the manual selective jump No. 3.

7. Optional Back Solution

After completion of the correlation problem proper, the machine will stop on a 56 00000 00430 command. If it is known at this time that a back solution is wanted, one will be obtained simply by pushing the start button. If this option is not exercised, and it is decided at some later time that a back solution is wanted, another program will permit computation of the back solution without repeating unnecessary parts of the correlation problem. A new code is read in; the same input is read in (with the x_j's either on cards or magnetic tape and the y's on cards) with the same control decks; and part of the card output of the correlation problem (the deck bearing the a's and the card bearing a_0) is also read in following the y's.

Machine Times

To date, two runs have been made in which accurate timing was recorded. Both were made on Univac Scientific Serial No. 9 at St. Paul, a magnetic core machine. The approximate formulas for computing machine time were derived from these two runs.

The first run consisted of a set of 16 problems run consecutively with the same set of independent variables. All short cuts were employed: the x's were read-in from magnetic tape, only the partials involving the dependent variable were computed, the inverse was not punched out, and no back solutions were obtained. The size of each problem was 17 variables, 91 observations. The average time per problem was one minute and 46 seconds, exclusive of code read-in time. Code read-in takes 50 seconds and, of course, need be done only once for any set of problems to be run at one time.

The second problem timed was one of 30 variables, the maximum number, and 95 observations. From the timing of this problem, times for the maximum case, 30 variables and 400 observations, can be closely estimated. The times quoted below for Sections 1, 2, and 6 are so estimated; those for Sections 3, 4, and 5 will be unchanged from those actually measured. All times are exclusive of code read-in time.

Maximum Case	Min	Sec	
1. Read all data from cards	8	50	
2. Means, standard deviation, r's	2	15	
3. Invert matrix	4	5	
4. Improve inverse	0	35	per cycle
5. Punch Decks 1 through 9 (full)	2	10	
6. Back solution - calc and punched	3	30	
Total - No Shortcuts	21	25	one imp. cycle

When the short cuts are used, the times will be about as follows for the maximum case:

Maximum Case	Min	Sec	
1a Read x's from tape, y's from cards	1	5	
5a Punch Decks 1-9 omitting Deck 8 and most of partials	0	50	
6a Omit back solution	0	0	
Total - All Shortcuts	8	50	one imp. cycle

Times for running problems of various sizes may be estimated from the following approximate formulas:

1. Reading all data from cards.

$$5 + (0.5)(m)(u+1) \text{ seconds}$$

where u is the number of cards required for n observations, at 12 to the card. This includes a fixed time of about 3.5 seconds for computing constants and typing the code and input sums and of 1.5 seconds for advancing the read cards and reading the control card.

1a. Reading x's from cards, y's from tape.

$5 + (0.5)(u+c) + (0.1)\left[(m-1)(v) + 2\right]$ seconds

where u is as above, c is the number of cards in the control deck, and v is the number of 32 word blocks required for n observations. This includes start-stop and computing time during the tape read-in.

2. Means, standard deviations, simple correlation coefficients.

$(0.012)(m)(n)$ seconds

This is quite rough, as there is a time involved that depends only on m and also one that depends on m^2n.

3. Invert matrix.

$(0.0091)(m^3)$ seconds

4. Improve inverse.

$(0.0013)(m^3)$ seconds per cycle

5. Compute and punch out Decks 1-9, full.

$(0.0044)(m^2)$ seconds for computing

$(0.25)(m) + (0.13)(m^2) + 3$ seconds for punching.

A more accurate estimate of card punching time can be made by computing the number of cards to be punched, from the information in "Card Output," and allowing 0.5 second per card.

5a. Optionally omitting partials not involving dependent variable and/or punch-out of inverse matrix.

$(0.030)(m^2)$ seconds saved by omitting extra partials

$(0.058)(m^2)$ seconds saved by omitting inverse

Again a more accurate estimate can be made by computing the exact number of cards saved.

6. Back solution.

$(0.0013)(m-1)(n)$ seconds for computing

$(0.5)(n) + 1$ seconds for punching

It is expected that these times will compare favorably with those of any existing multiple regression and correlation program.

Input - Output

Scaling

All input data should be scaled so that the five most significant digits are to the left of the decimal point, as the input is in the form of integers of five digits or less. It is desirable, but not absolutely necessary, that the observations of a given variable be scaled so that the largest (in absolute magnitude) has its leading significant digit appearing in the fifth place to the left of the decimal. However, if convenience dictates, all the observations of a variable considered as five digit integers may contain zeros in the leading (or possibly even the leading two) digits. There is danger, however, that if the dependent and independent variables are scaled to give numbers of very dissimilar magnitudes, considered as integers, some of the regression coefficients may be so small as to lose some significant figures or so large that they overflow, in which case the problem must be rescaled to be run. It is therefore recommended that each variable be scaled so that the largest observation encountered (or likely to be encountered in subsequent problems) have its leading significant digit in the fifth place to the left of the decimal. The variables so scaled are now the input variables for the problem, as far as the machine is concerned, and the output will come out scaled accordingly and correctly pointed off with actual printed decimal points.

Paper Tape Input

The instruction code for the correlation program, with optional back solution, is contained on a single roll of paper tape, requiring about 50 seconds to read. The code for the back solution alone (correlation done previously) is on a shorter single roll. Both codes are completely self-contained and need no other routines or subroutines except to load them. Both are in bioctal code, and both may be read with either an ERA or a Ferranti reader. The Ferranti Load Routine RW 63, operating from 75170 to 75337, has been used. The code occupies drum addresses 40000 through 43000 and 77074 through 77777; a load routine must be used that permits loading to these addresses.

Card Input

The input data are read into the machine on cards. As stated before, if one set of independent variables is to be used for a number of problems, it may be written on magnetic tape as it is read in from cards initially, and in subsequent problems only the dependent variables need be read in from cards. In either case, a control deck is first read in, followed immediately by a deck of cards for each variable. The latter decks have the following format. There are 13 fields on each card, starting in Column 3, each of five columns followed by a sign column. The first field contains the code number which identifies the variable. This code number may be any integer of five digits or less, except that code numbers for dependent variables should be positive. The following 12

fields of the first card in the deck contain the first 12 observations of that variable, the corresponding 12 fields of the second card the next 12 observations of that variable, etc. The observations are expressed as five digit numbers (see "Scaling") followed by a blank column if the number is positive or a column containing only an x (11) punch if the number is negative. It is not necessary to punch all columns of the card, but any blank column except the sign columns will be interpreted by the 1103 as a zero; thus any zero in any data word may be either punched or left blank. Unless the number of observations is an even multiple of 12, the last card in each deck will not be filled; in this case the remaining fields may be left blank or used for any other purpose, as it will make no different at all what is in them. Since it is imperative that all the cards in each deck remain in the proper order, the first two columns on the card, which are not read by the 1103, should contain the number of the card in the deck for sorting purposes in case of mixup.

There must be a full set of observations for all independent variables, as any blank space will be interpreted an observation of value zero. However, when a number of dependent variables are run with one set of independent variables, there may be missing observations in the dependent variable. These should be designated by a minus zero (a field of blank or zero columns followed by an x punch in the sign column) in the proper place on the card. A plus zero or

completely blank field will be interpreted as a legitimate observation of value zero. The machine will distinguish between minus zero and zero only in the dependent variables; a minus zero in an independent variable will be interpreted as an ordinary zero. When a dependent variable with certain observations missing is used in a given problem with a full set of independent variables, "n," the number of observations for the control card (see below) is taken as the full number of observations of the independent variables. The machine will scan the list of observations of the dependent variables for minus zeros, strike out the corresponding observations of the independent variables, and subtract 1 from "n" for every missing observation.

When all data are to be read from cards, the control deck consists of a single card with "m" the number of variables punched in Columns 10 and 11, and "n" the number of observations punched in Columns 14, 15, and 16. It will make no difference what else, if anything, is on the card from Column 18 on. This control card is immediately followed by the decks for the independent variables, which are again immediately followed by the deck for the dependent variable. It is necessary that the dependent variable be read in last (even though it will be treated as the first variable in the matrix and will assume the leading position once in the drum memory), as that is the only identification of it as being dependent. The independent variables, however, may be read in in any order, so long as the cards within each deck remain in order. The first output from

the problem will consist of the identification numbers of all the variables, starting with the dependent variable and followed by the independent variables in the order which they were entered; subsequent output will follow the same order, so that the order in which the independent variables are entered will be recorded and preserved in the output.

When the independent variables are to be read from magnetic tape, the first card in the control deck will contain, in Columns 1 and 2, the number of variables to be omitted, in case it is not desired to use all the variables in the correlation. (If none are to be omitted, a single blank card will serve as the control deck.) This card will be followed by as many cards as there are variables to be omitted, each card bearing the code number of a variable to be omitted (in any order) in Columns 3 through 7, with Column 8 as a sign column. For example, if the code number of a variable to be omitted is 13, this number will be punched in Columns 6 and 7; if it is minus 98745, 98745- will be punched in Columns 3 through 8. The machine will not read anything on these cards from Column 9 on. This control deck will be immediately followed by the deck of observations of the dependent variable. This system is used because when a set of independent variables is already on tape, there is also on tape m and n for the general case: all variables used, all observations present. The control deck will cause m to be adjusted for the particular problem, and minus zeros in the dependent variable observations will cause n to be adjusted.

In any problem in which cards are used as input, it is necessary that at least three blank cards follow the last card to be read. If several problems are to be run at once, two blank cards must follow each problem and at least three must follow the last problem. In some cases it is desirable to run a number of problems at a time, each with a different dependent variable but all with the same set of independent variables. In this event, all data are read from cards for the first problem, with exactly two blank cards following the deck for the dependent variable. Then follows the control deck (which may be one blank card) for the second problem, immediately followed by the dependent variables, followed by two blank cards, etc. In this way, all the problems can be loaded at once with no card handling on the input side after the machine has started. The independent variables, of course, are written on tape during the first problem and are read from tape in subsequent problems.

When the correlations have already been done and only a back solution is required, the independent variables may be read in from either tape or cards as in the correlation problem. The control decks and the card input for the variables are exactly as before. The deck for the dependent variable is followed in this case by a deck bearing the regression coefficients, a_j, (deck 5 of the output) and then by a card bearing (among other things) the constant term in Equation (1), a_0, (deck 9 of the output). Again, this card is followed by at least three blanks, or exactly two blanks if another problem

is to follow immediately. These decks 5 and 9 will ordinarily be the identical cards produced by the machine when the correlation phase of the problem was performed; their format will be described under "Card Output."

Typewriter Output

Although the bulk of the output is on cards, the typewriter is used to some extent as a monitor as the problem runs.

The first act of the machine in starting the problem is to sum the code on the drum and write the sum on the typewriter. It would not be advisable until experience has demonstrated conclusively that no more changes in the code are desirable to quote this sum here or to build in an automatic check. In the meantime, the fact that the code is properly in the machine can be verified by the appearance of a familiar number as the first line of typewriter output.

The second possible item of typewriter output will appear only if the independent variables are read in from tape and if their sum as read in fails to agree with their sum written on the tape. In that event, there will appear on the next line "tape sum no good" or a suitable abbreviation thereof. (All variables on the tape are summed as read, whether or not the control deck indicates they are to be used in this problem, so the sum should always be the same.) If this appears, the machine will stop on a 56 00000 40576 command Pushing the start button will cause it to attempt to read the tape again;

a failure repeated several times will indicate mechanical difficulty in the tape or in the reading circuits.

The next item of typewriter output, on the following line, will be the sum of all input data, including the dependent variables. This will normally vary from problem to problem and aside from inspection of the general magnitude will serve as a useful check only when difficulty is suspected and it is desired to repeat the problem and test for trouble. However if only the back solution is to be run, this input sum should check that written out in the correlation phase of the problem.

The next item of typewriter output, on the following line, will be the letter i written one or more times. Each i signifies the completion of the first half of a cycle in the improvement scheme for the inverse matrix. If the improvement scheme converges, the carriage will return and the typewriter will print the next output. If it diverges, it will write, on the same line, "diverges. set mj 2,3. go," after which the machine will stop on a 56 00000 00132. Pushing the start button will bring back the original inverse as it was before the start of the improvement scheme, and the machine will continue the problem using that. However, in case of divergence it is likely that the user will want full information to indicate why this occurred. The "set mj 2,3" is a reminder to set these manually selected jumps, if they are not already set, before pushing the start button if it is desired to see the

print out of all the partial correlation coefficients (2) and the inverse of the correlation matrix (3). (As indicated before, the code has room for another and more powerful improvement scheme that can be applied if this fails to converge. If this section is shown to be desirable and is written, failure of the first method to converge will not activate the typewriter, but failure of both methods will cause it to write "matrix sing.")

The final item of typewriter output will be the square of the multiple correlation coefficient. This is written as soon as computed because sometimes problems may differ only in that alternate functions of the variables are being tried (linear, logarithmic, higher order algebraic functions of the basic data), and when this is the case for the dependent variable, the choice of the function giving better fit can be made solely on the basis of the multiple correlation coefficient. If it turns out that the problem now in the machine gives a coefficient lower than that for an alternate form previously run, the problem may, if desired, be abandoned immediately by forcing a stop at a saving of machine time.

Card Output

The card output is designed to be listed on a Type 407 tabulator using a straight 80-80 board, all necessary spacing being done on the cards themselves. All numbers will be properly pointed off with decimal points (8-3-12 punches on the cards) and will be followed by minus signs when negative (x or 11 punch). It is also possible to list the output on a

402 tabulator without total loss of legibility, but this is awkward because the fields are not located in the same places in all cards. In using a 402, a straight 80-80 board is also used, except that there must be provision for recognition of minus signs, and it is better to use zero suppression on the right-hand side of the page. Minus signs are possible in Columns 16, 20, 24, 32, 35, 40, 47, 48, 50, 52, 56, 62, 64, 65, 72, 77, and 80. In the 402 listing, decimal points will appear as 8's (or as 9's in columns where a minus sign can occur), and minus signs will appear as a 9 or some other symbol, but the output is so organized on the page that it will not be difficult to distinguish these symbols from the legitimate numbers 8 and 9 in data words.

The output is in nine decks, exclusive of the back solution. In the first column of each card is punched the number of the deck, and in the next two is the number of the card in the deck. Each deck will be briefly described below, with the fields located with respect to the decimal point. The number of columns mentioned to the right of the decimal point is exclusive of the sign column, which is always the right-hand column of the field.

Deck One

The code numbers of the variables. Dependent variable first, independent variables in the order in which they were read in. Five numbers to a card. No decimal points printed, as all numbers are integers, but locations of hypothetical decimal points are in Columns 17, 32, 47, 62, and 77.

Decks Two through Five

The means, standard deviations, normalized regression coefficients (the b_j's of Equation 9), and the regular regression coefficients, respectively. Five numbers to a card. All variables in the same order as the code numbers, and all lined up directly below their respective code numbers on the page. Decimal points in Columns 14, 29, 44, 59, and 74. Five digits to each side of decimal.

Decks Six and Seven

The simple and partial correlation coefficients, respectively. Nine numbers on a card. Decimal points in Columns 10, 18, 26, 34, 42, 50, 58, 66, 74. No places to the left of the decimal, five to the right. As the matrix of coefficients is symmetrical with diagonal elements unity, only the upper right triangle without the diagonal is reproduced. The first row of the triangle starts with the coefficient of correlation between the dependent variable and the first independent; succeeding elements are the coefficients between the dependent variable and the second, third, etc., independent variables, in the order in which they were read in. The second row of the triangle starts with the coefficient between the first and second independent variables, and continues with the coefficients between the first and third, fourth, etc. Each row is one shorter than the one before. The elements of the first row are punched on cards nine at a time; when there are no longer nine to punch, the remaining spaces on the card are filled with zeros. This process is repeated with

the next row of the triangle. Thus the completion of each row of the triangle is signified by one or more words of .00000 appearing to the right of a line on a page, except for those rows containing a number of elements that is an exact multiple of nine. In this way, the row of the triangle is easily identified in the printed output. The punching out of more than the first row of the matrix of partial correlation coefficients is optional.

Deck Eight (Optional)

The inverse of the correlation matrix. Five numbers to a card. Decimal points in Columns 13, 28, 43, 58, 73. Four places to the left of the decimal, six to the right. As the inverse matrix is symmetrical, the same scheme is used here as with Decks 6 and 7, except that the diagonal is included, so that each row is one longer than the corresponding row of the matrix of correlation coefficients. Again, the completion of the row of the triangular matrix will be signified, except for those rows containing a number of elements that is a multiple of five, by one or more words of 0.000000 appearing to the right of a line on the page.

Deck Nine

Deck nine consists of one card bearing the square of the multiple correlation coefficient, the standard error of estimate, and a_0 of Equation (1). The decimal point for the first is in Column 15; no places to the left, five to the right. The decimal point for the second is in Column 43; five places to each side. The decimal point for a_0 is in Column 77; eight places to the left, two to the right.

The Back Solution (Optional)

The cards in the back solution each bear four numbers. The first number on the card is a nine-digit integer starting in Column 1. The first five digits of this word are the code number of the dependent variable, the next digit is zero, and the last three digits are the number of the observation of the dependent variable dealt with on this card. One card is punched for each observation. The next number in the card (decimal point in Column 21, eight places to the left, two to the right) is the value of y_h^* computed by Equation (1). The next number (no decimal point punched, would be in Column 35, five places to the left) is the observed value of y for observation number h, y_h. The last number (decimal point in Column 49, eight places to the left, two to the right) is the difference $y_h^* - y_h$.

Operating Instructions - Full Correlation Problem

The operating instructions will be given a sequence designated by Roman numerals. This sequence is based on the assumption that no abnormal situations or mishaps will occur at any stage in the problem. However, there are certain check points in the program at which any errors or abnormal situations can be detected, and specific operating procedures have been devised in case of an abnormal situation at any check point. These check points will be designated by asterisks between the lines of the normal sequence, and the procedures devised for each point will be given following the normal sequence.

I. Initial Preparation

A. Set Card Unit Field III switch to Normal.

B. Clear the read channel and place cards, prepared as in "Card Input," in hopper.

C. Turn on MT 1 and MT 2 and position the tapes.

D. Turn the typewriter on.

E. Use a 4-interlace for maximum efficiency.

II. Load the Program

III. Master Clear

A. Clear and MD start.

B. Put computer on High Speed, Test Mode, Drum Clock Source.

IV. Make Manual Jump Selections

A. No. 1 if x's are to be written on tape.

B. No. 2 if all partial correlation coefficients desired.

C. No. 3 if inverse of correlation matrix is to be punched out.

V. START the computer. The typewriter will print out the code sum and the computer will halt on a 56 00000 40100 instruction. Then:

A. If the independent variables are to be read from cards, START the computer.

B. If the independent variables are to be read from tape, set PAK=00000 and START the computer. MT 2 will be used. In either case, the computer will:

1. Advance the cards two stations and read control deck.
2. Read x's from cards or tape.

*

3. Read y's from cards.
4. Write input sum on typewriter.
5. Compute means, standard deviations, cross products (brief, characteristic scope display for each).
6. Invert the correlation matrix (spectacular scope display).

**

7. Start improving matrix, using MT 1 if there are more than 19 variables (scope display characteristic of first half of improvement cycle).
8. Type out "i" after first half of each improvement cycle.

***

9. Finish improving matrix (scope display characteristic of second half of improvement cycle) by repeating Steps 7, 8, and 9 as often as necessary. Frequently, once is enough.
10. Type out the square of the multiple correlation coefficient.
11. Compute standard error of estimate, normalized and regular regression coefficients, and partial correlation coefficients (very brief scope display).
12. Advance punch cards two stations and punch out decks 1 through 9.

****

13. STOP on a 56 00000 00430 instruction.

VI. Back Solution after Correlation (Optional)

A. If back solution is <u>not</u> desired, problem is finished. If another problem is ready in the card read hopper (see "Card Input" for directions for loading several problems at one time), master clear, MD start, and go to IV.

B. If back solution <u>is</u> desired, START the computer. It will:

1. Compute back solution (very brief characteristic scope display).
2. Advance punch cards two stations and punch out out back solution.

*****

3. FINAL STOP on a 57 00000 00000 command.

C. If back solution was run and another problem is ready in the hopper, to to III.

* Trouble Symptom - Typewriter writes "tape sum no good," or a suitable abbreviation thereof, and computer halts on a 56 00000 40576 command.

Action - START computer. Computer will attempt to read tape again. If it succeeds, it will proceed without comment. If it fails, the above symptom will be repeated.

Explanation - In reading the x's from tape, the computer will sum the data as read in and compare this sum with one written on the tape. If the sums agree, the computer will rewind the tape while proceeding with the problem. If they do not, the computer will rewind the tape while typing out the message and then halt. If the fault is caused by a dropped bit, a second reading may be successful, but this may indicate the tape needs replacement or regeneration.

** Trouble Symptom - The computer halts, right after the highly characteristic scope display indicating the matrix inversion, on a 56 00000 00010 command.

Action - Master clear, MD start, set PAK = 42524, and START. The computer will advance the punch cards two stations, punch out decks 1, 2, 3, and 6, and STOP on a 57 00000 00421 command.

Explanation - If the matrix is to be inverted is singular, or nearly singular, the computer may be unable to invert it because of overflow in the floating point arithmetic or, as is more likely, because of overflow in converting the elements of the inverse matrix back to fixed point numbers. Such an overflow will cause the above halt. Taking the above action will cause the computer to print out all the results obtained up to this point in the problem, after which it will come to a final stop, as nothing more can be done with this problem. The cause for the singularity of the matrix can generally be found in the magnitude of the simple correlation coefficients; if the explanation is not there, it may be due to $n \leq m$, or to a nearly perfect fit of the regression plane.

*** Trouble Symptom - In the matrix improvement, the scope display characteristic of the first half of the cycle reappears after the typewriter types "i." The typewriter then types "diverges. set mj 2,3. go," and the computer then halts on a 56 00000 00132 command.

Action - Set manual selective jumps 2 and 3 if all the partials and a listing of the inverse matrix are desired. Then START. The computer will finish the problem in the normal manner.

PX 71900-9-(127)

Explanation - The matrix improvement scheme has failed to converge, probably indicating a poorly conditioned matrix. The computer halts to give the operator the opportunity to set these manually selective jumps in case full information on the inverse is desired because of this condition, but it is not imperative to set them. Starting the computer will restore the inverse as obtained directly, before the initiation of the improvement scheme, and the problem will be continued with this. Divergence is usually caused by the original inverse being not quite good enough to meet the tolerance limit imposed but still so good that the small improvement that can be effected in one cycle becomes smaller than accumulated error in the improvement scheme.

**** Trouble Symptom - IO fault, other fault, or other evidence of difficulty with card unit.

Action - Clear fault. Force stop. Check condition of card unit, cards. Remove frayed or warped cards from the punch hopper. Clear the punch channel and remove punched cards. See that there is an adequate supply of well conditioned cards, not stuck together, in the feed hopper. Then master clear, MD start, see that the machine is on High Speed, Drum Clock Source, and Test Mode. Set PAK = 41310 and START. Machine will recommence punching out results starting with deck 1.

PX 71900-9-(127)

Explanation - During the punching out of the answers, trouble can arise in the card unit because of poorly conditioned cards, because too many punched cards have accumulated in the output hopper, or for some other reason. Since the answers are preserved on the drum, it is possible to start over in the punch-out routine once the cause of the difficulty has been corrected.

***** Trouble Symptom - Same as for **** above, except that it occurs in the punching out of the back solution.

Action - Same as above, except that PAK is set equal to 42114.

Explanation - Same as above.

OPERATIONS RESEARCH OFFICE
7100 Connecticut Avenue
Chevy Chase, Maryland

Complab

Coded by S. Rigby and J. Chappell

Page 1 of 15

Checked by S. Rigby and J. Chappell

Date 3 May 1956

Computer checked by S. Rigby and J. Chappell

Title: Magnetic Drum to Magnetic Tape Dump

Use: This routine is used to dump specified portions of the magnetic drum onto the magnetic tape units. Returning of the information to the drum is under control of a routine which is written on the tape at the time of the dumping.

Range: Any one of the tape units may be used and any number of words from 1 to the entire drum may be dumped. Upon restoration of the information to the drum, only the exact words dumped are restored; the remainder of the drum is undisturbed.

Storage: Initial storage of the routine is:
00100b - 00161b
00200b - 00540b
The entire magnetic core memory is used for temporary storage.

Format: This routine is not coded in standard form, cannot be modified, and is not self-resetting.

Parameter Words: Parameter words for the control of this routine are placed in A and Q. See Instructions for Use, page 3.

Manual Entry: 00100b

Automatic Entry: For use with Ferranti Read-in routines recognizing a transfer to program code, an automatic start at 00105b is available.

PX 71900-9-(128)

Description of Service

This routine is used to dump information from the magnetic drum onto the magnetic tapes. The information is moved from the drum into the magnetic core before being written onto the tape. The entire core is used for this operation. After a start the routine first writes 4 blocks of tape which contain the routine for restoring the information to the drum. The desired information is then written onto the tape, containing as the first word of the first block of information, a control word which contains the insert address for the information and the number of locations. By resetting the parameter word in Q, additional groups of information can be dumped, each with its own control word. By a restart with Q = 0, the signal that the last group has been written, one additional block is written which contains a "Dump end" control word and a check sum of all dumped information. The magnetic tape is then rewound to the original starting position and the computer stops. All control words contain a parity bit which will be a 0 or a 1 such as to make the number of 1's in every control word even.

Provision is made for the inclusion of an address to which control of the computer will be returned after the dumped information has been restored to the drum from the tape. If such address is used, the computer will also come to a zero stop with PAK set at this address after the dumping has been completed and the tape rewound. At this time, the total number of blocks that have been written will appear in Q.

To restore the information on the tape to the drum, an MT Start is used with the appropriate J in VAK and the tape unit positioned. The undump routine is first read into core and a check sum taken on the undump routine itself. Next the block containing the control word is read in and the control word itself is subjected to a parity check. The information is then read into core and transferred to the appropriate location in drum, where a check sum on the information is computed. The magnetic tape is then rewound and control returned to the address designated when the information was dumped. If no such address was designated, a FS results.

Instructions for Use of Routine

Dumping from Drum to Tape

1. Manually dump MC to MD, if desired.

2. Position the magnetic tape which is being used to the position where dumping is desired.

3. Enter parameter word in Q:

   OJ VVVVV NNNNN

   J = desired tape unit
   V = First Drum address of information being dumped
   N = number of words to be dumped

   If V + N exceeds 100000b, N is replaced by 100000b-V (i.e., last word dumped is 77777)

4. Enter in A_R the following:

   00 00000 BBBBB

   B = MD address to which control of machine should be returned after information on tape has been restored to drum, if any. This may be omitted if a Final stop is desired after information is read from tape onto drum.

5. Set PAK to 00100 and Start.

6. If MT0 has been put in the Q parameter word or if no tape unit has been specified, (i.e., J=O), the typewriter will type out "mt0" and stop. If unit zero is desired, simply restart. If one of the other tape units is desired, insert the correct J in Q and restart. If no parameter word was inserted in Q before starting, the typewriter will type out "set q".

7. After dumping the information designated by the first parameter word, the machine will halt with Q clear. At this time additional

groups of information may be dumped by keying into Q additional parameter words and restarting. The parameter word in A_R should not be used after the first dump. As many groups of words may be dumped as desired.

8. After the last desired group of words has been dumped, the machine should be restarted with Q = O. This will signal the routine that the dumping is finished; the final control word and the check sum will be written and the tape rewound. At this point the machine will come to a 0 stop with PAK equal to the B address inserted in A_R in the original dump; if such was used. If no parameter word was placed in A_R, a Final Stop results at this point.

9. At this time, the total number of blocks of tape that have been written will appear in Q for logging purposes.

Restoring Information to Drum

With the tape unit positioned to the same place where the dumping began, and the appropriate tape unit keyed into UAK, an MT Start should be made. The information dumped will be stored on the drum, check sum compared, tape rewound, and, if no B address was used in A_R during the initial dump, come to a final stop. If a B address was used in A_R during the initial dump, control is sent to B via an MSI.

Alarms and Abnormal Conditions

1. If no tape number is keyed into the parameter word, the Flexowriter types a warning to the effect, but a restart will permit using Tape unit zero.

2. If any address other than a drum address is set for any V, the routine will not dump the information but will repeatedly return to a zero stop.

3. If an address other than a drum address is used as B for the start of the program, a Final Stop will result at the end of both the dump and the undump.

4. If V + N exceeds 100000b, N is replaced by 100000b-V.

5. When restoring the information from tape to drum, the undump routine first performs a check sum on itself. If this fails, the typewriter types out "Undump Check Sum NG". A restart reads in the routine again and recomputes the check sum.

6. The first word of each section restored to drum is a control word containing the insert address and number of words in that section and a parity bit such as to make an even number of 1's the word. Each control word is checked by the routine for this feature. Should this fail, the Flexowriter types out: "Parity check failure". A restart will repeat the read-in and parity check.

7. After all information has been read back into the drum, a low order check sum of all data is made and an error causes the typewriter to type out "Data check sum fault". A restart reads the data in again.

SETUP ROUTINE

Storage Location	Working Location				
00100	00100	45	00000	00101	Manual Entry
00101	00101	11	20000	01001	Store B, if used
00102	00102	11	10000	01000	Store J, V, and N
00103	00103	11	10000	20000	Test to see if parameter
00104	00104	47	00113	00106	word was placed in Q
00105	00105	23	10000	10000	Auto start; clear Q
00106	00106	31	00153	00052	
00107	00107	61	00000	20000	
00110	00110	34	20000	00006	Print "set q"
00111	00111	47	00107	00112	
00112	00112	56	00000	[00101]	
00113	00113	31	01001	00071	Test to see if B is MD
00114	00114	46	00115	00117	address
00115	00115	11	00112	00521	
00116	00116	45	00000	00120	
00117	00117	11	00521	00201	Set up exits
00120	00120	16	01001	00521	
00121	00121	16	01001	00201	
00122	00122	11	01000	10000	Parameter word → Q
00123	00123	51	00155	20000	J → A
00124	00124	47	00127	00125	J = 0 ?
00125	00125	31	00154	00052	Print "mt0"
00126	00126	37	00112	00107	
00127	00127	55	10000	00041	Shift tape unit number
00130	00130	51	00355	00356	J → 00356b
00131	00131	11	00355	10000	J mask → Q

```
00132    00132    11  00100  20000
00133    00133    42  00156  00135
00134    00134    42  00157  00140
00135    00135    21  00132  00372
00136    00136    42  00160  00132
00137    00137    45  00000  00144
00140    00140    31  00132  00071
00141    00141    16  20000  00142
00142    00142    53  00356  00000
00143    00143    45  00000  00135
00144    00144    75  10174  00146
00145    00145    21  00375  00201
00146    00146    13  00375  00375
00147    00147    65  00004  00200
00150    00150    11  01000  10000
00151    00151    75  30137  00001
00152    00152    11  00401  00001
00153    00153    45  24200  10435
00154    00154    07  01025  73702
00155    00155    03  00000  00000
00156    00156    63  00000  00000
00157    00157    67  70000  00000
00160    00160    11  00600  00000
```

Routine to mask tape unit number into all tape orders

Compute check sum of undump routine

Write undump routine on tape unit J

Parameter word → Q

Move dump routine to 00001 - 00140b

Constants

PX 71900-9-(128)

UNDUMP ROUTINE

00200	00000	45	00000	00002	
00201	00001	56	10000	[00000]	Exit to main program
00202	00002	64	00003	00040	Read in remainder of undump routine
00203	00003	75	10174	00005	Test check sum of undump routine
00204	00004	21	00175	00001	
00205	00005	47	00007	00014	
00206	00006	00	07777	00000	Constant - block mask
00207	00007	67	00004	00000	Rewind tape for new read-in if check sum fails
00210	00010	15	00013	00132	Print out "undump check sum NG"
00211	00011	37	00137	00130	
00212	00012	64	00004	00000	Read in undump routine again when check sum fails
00213	00013	45	00140	00003	
00214	00014	15	00063	00053	Repair 00053b
00215	00015	64	00001	00177	Read in block containing control word
00216	00016	23	00237	00237	Parity check on control word
00217	00017	11	00173	00171	
00220	00020	55	00177	00001	
00221	00021	11	00237	20000	
00222	00022	52	00172	00237	
00223	00023	41	00171	00020	
00224	00024	31	00237	00070	
00225	00025	46	00026	00033	
00226	00026	67	00001	00000	
00227	00027	41	00174	00015	
00230	00030	15	00032	00132	Print out "Parity check failure"
00231	00031	37	00137	00130	

PX 71900-9-(128)

00232	00032	45	00144	00015	Return to read in control word again
00233	00033	11	00177	20000	Check for final control word
00234	00034	46	00044	00035	
00235	00035	15	00177	00036	Rewind tape
00236	00035	67	[00000	00000]	
00237	00037	11	00200	20000	Test check sum, exit to main routine if ok
00240	00040	43	00176	00001	
00241	00041	15	00043	00132	Print "Data check sum fault"
00242	00042	37	00137	00130	
00243	00043	45	00150	00012	
00244	00044	15	00177	00170	N → 00170b
00245	00045	16	00177	00053	Setup v of 00053b
00246	00046	11	00170	20000	N → A
00247	00047	42	00166	00110	N > 37 ?
00250	00050	42	00167	00104	N > 1577 ?
00251	00051	64	00033	00237	Read in 1540 words
00252	00052	75	31577	00054	Transfer 1577 words to MD
00253	00053	11	00200	[00000]	
00254	00054	37	00126	00120	Compute check sum
00255	00055	21	00053	00165	v + 1577 → v
00256	00056	23	00170	00164	N - 1577 → N
00257	00057	42	00163	00070	N > 1600 ?
00260	00060	64	00034	00200	Read in 1600 words
00261	00061	16	00053	00063	Pick up current v
00262	00062	75	31600	00064	Transfer 1600 words to MD
00263	00063	11	00200	[00000]	

PX 71900-9-(128)

00264	00064	37	00126	00120	Compute check sum
00265	00065	21	00053	00162	V + 1600 → V
00266	00066	23	00170	00161	N - 1600 → N
00267	00067	45	00000	00057	
00270	00070	21	20000	00160	N + 37 → A
00271	00071	73	00157	20000	$\frac{N+37}{40}$ → A
00272	00072	21	20000	00156	J; $\frac{N+37}{4}$ → 00074b
00273	00073	15	20000	00074	
00274	00074	64	00000	00200	Read in $\frac{N+37}{40}$ blocks
00275	00075	15	00155	00100	
00276	00076	21	00100	00170	N → 00100b
00277	00077	16	00053	00101	
00300	00100	75	[00000]	00102	Transfer N words to MD
00301	00101	11	00200	[00000]	
00302	00102	37	00126	00120	Check sum
00303	00103	45	00000	00014	Read in next section of dumped information
00304	00104	73	00157	20000	$\frac{N}{40}$ → A
00305	00105	35	00156	20000	
00306	00106	15	20000	00107	J; $\frac{N}{40}$ → 00107b
00307	00107	64	[00000]	00237	Read in $\frac{N}{40}$ blocks
00310	00110	11	00170	20000	
00311	00111	35	00155	20000	J, N → 00114b
00312	00112	15	20000	00114	
00313	00113	16	00053	00115	Pick up V
00314	00114	75	[00000]	00116	Transfer N words to MD
00315	00115	11	00200	[00000]	
00316	00116	37	00126	00120	Compute check sum
00317	00117	45	00000	00014	Read in next section of dumped information

PX 71900-9-(128)

00320	00120	11	00126	20000	Check sum routine
00321	00121	34	00166	00017	
00322	00122	15	20000	00124	
00323	00123	11	00006	10000	
00324	00124	53	[00000]	00126	
00325	00125	16	00053	00127	
00326	00126	75	[10000]	[00000]	
00327	00127	21	00176	[00000]	
00330	00130	16	00166	00171	Print subroutine
00331	00131	16	00003	00174	
00332	00132	55	00000	00006	
00333	00133	61	00000	10000	
00334	00134	41	00174	00132	
00335	00135	21	00132	00172	
00336	00136	41	00171	00131	
00337	00137	56	00000	[0000]	
00340	00140	45	47345	70622	Flex code constants
00341	00141	34	07150	41605	
00342	00142	20	16360	42434	
00343	00143	07	04470	61345	
00344	00144	45	47155	73012	
00345	00145	14	01250	41605	
00346	00146	20	16360	42630	
00347	00147	14	11341	22045	
00350	00150	45	47225	73001	
00351	00151	30	04160	52016	
00352	00152	36	04243	40704	
00353	00153	26	30341	10145	

PX 71900-9-(128)

00354	00154	00	00000	00003	Constants and Counters
00355	00155	00	30000	00000	
00356	00156	00	00000	00000	
00357	00157	00	00000	00040	
00360	00160	00	00037	00000	
00361	00161	00	01600	00000	
00362	00162	00	00000	01600	
00363	00163	00	01600	00001	
00364	00164	00	01577	00000	
00365	00165	00	00000	01577	
00366	00166	00	00037	00003	
00367	00167	00	01577	00001	
00370	00170	00	[00000]	00000	
00371	00171	00	00000	[00000]	
00372	00172	00	00001	00000	
00373	00173	00	00000	00043	
00374	00174	00	00000	[00001]	
00375	00175	00	00000	00000	
00376	00176	00	00000	00000	
00377	00177	00	00000	00000	

DUMP ROUTINE

00400	00000	45	00000	00001	
00401	00001	31	10000	00052	Parameter word → A
00402	00002	47	00014	00003	Parameter word zero ?
00403	00003	23	00137	00137	Make control word positive ?
00404	00004	15	00012	00137	J; N → control word
00405	00005	37	00040	00027	Parity bit correction
00406	00006	11	00136	00140	Move check sum
00407	00007	65	00001	00137	Write control word and check sum on tape

PX 71900-9-(128)

00410	00010	11	00127	10000	Number of blocks → Q
00411	00011	51	00012	10000	
00412	00012	67	[00005]	00043	Rewind tape
00413	00013	45	00000	00120	
00414	00014	46	00015	00104	Is V an MD address ?
00415	00015	15	10000	00051	} Store V
00416	00016	31	10000	00071	
00417	00017	16	20000	00137	
00420	00020	31	10000	00017	} Store N
00421	00021	15	20000	00130	
00422	00022	31	00135	00065	100000b → A
00423	00023	16	00137	00133	Isolate N
00424	00024	34	00133	00017	100000 - N to Au
00425	00025	42	00130	00021	V + N > 100000b ?
00426	00026	15	00130	00137	N → control word
00427	00027	16	00012	00133	00043b → counter
00430	00030	23	00237	00237	0 → parity sum
00431	00031	27	00137	00135	Complement parity bit
00432	00032	11	00137	10000	Control word → Q
00433	00033	55	10000	00001	Shift Q left 1 place
00434	00034	11	00237	20000	} Add 1 bit of control word to parity sum
00435	00035	52	00132	00237	
00436	00036	41	00133	00033	All bits added ?
00437	00037	31	00237	00070	Parity sum → A
00440	00040	46	00027	00041	Least significant bit of parity sum zero ?
00441	00041	11	00130	20000	N → A
00442	00042	15	00125	00124	1637b → 00124b
00443	00043	15	00065	00050	Setup transfer of 1637b words

00444	00044	16	00004	00053	Setup to write control word
00445	00045	11	00125	00131	Setup index to 1637b
00446	00046	15	00064	00134	Setup block count threshold to 40
00447	00047	42	00124	00067	N less than 1637b or 1640b ?
00450	00050	75	[00000]	00052	Transfer 1637b or 1640b words
00451	00051	11	[30000]	00140	
00452	00052	37	00116	00105	Compute check sum
00453	00053	65	[0035]	[30000]	Write words on tape
00454	00054	21	00012	00123	35 plus block count to block count
00455	00055	21	00051	00131	V + 1637b or 1640b → V
00456	00056	23	00130	00131	N - 1637b or 1640b → N
00457	00057	15	00126	00124	Set threshold to 1640b
00460	00060	15	00122	00050	Set transfer order to 1640b words
00461	00061	16	00051	00053	Set "v" of tape write order to 140b
00462	00062	11	00126	00131	Set index to 1640b
00463	00063	15	00104	00134	Set block threshold to 37
00464	00064	45	00040	00047	
00465	00065	00	31637	00000	
00466	00066	00	00000	00000	
00467	00067	32	00134	00103	$\frac{N + 37}{40} \rightarrow A$
00470	00070	11	00127	10000	N mask → Q
00471	00071	53	20000	00102	Number of blocks → write mtj order
00472	00072	53	00130	00074	N → 00074b
00473	00073	15	00051	00075	V → 00075b
00474	00074	75	[30000]	00076	} Transfer remaining words in section to MC
00475	00075	11	[30000]	00140	
00476	00076	37	00116	00105	Compute check sum
00477	00077	16	00053	00102	Pick up start point from mtj
00500	00100	11	00012	20000	} Block count
00501	00101	52	00102	00012	

PX 71900-9-(128)

00502	00102	65 [00000] [30000]	Write mtj
00503	00103	23 10000 10000	Clear Q
00504	00104	56 00037 00001	Wait for next parameter word
00505	00105	11 00116 20000	Setup check sum routine
00506	00106	34 00124 00017	V to 00117v
00507	00107	15 20000 00110	
00510	00110	31 [30000] 00071	
00511	00111	16 20000 00117	
00512	00112	23 00110 00132	Y - 2 → 00115v
00513	00113	15 20000 00115	
00514	00114	11 00127 10000	N mask → Q
00515	00115	53 [30000] 00116	Number of words transferred → 00116v
00516	00116	75 [10000] [30000]	Compute deck sum
00517	00117	21 00136 30000	
00520	00120	55 10000 00025	Shift number of blocks to QV
00521	00121	[57 00000 00000]	Stop (or 5600000 BBBBB)
00522	00122	00 31640 00000	Constants and Counters
00523	00123	00 00035 00000	
00524	00124	00 [00000] 00002	
00525	00125	00 01637 00000	
00526	00126	00 01640 00000	
00527	00127	00 07777 00000	
00530	00130	00 [00000] 00000	
00531	00131	00 [00000] 00000	
00532	00132	00 00001 00000	
00533	00133	00 00000 [00000]	
00534	00134	00 [00000] 00000	
00535	00135	20 00000 00000	
00536	00136	00 00000 00000	
00537	00137	40 [00000] [00000]	

PX 71900-9-(128)

ANALYSIS
PREPARED BY Hauser and Gerkin
CHECKED BY
REVISED BY

CONVAIR
A DIVISION OF GENERAL DYNAMICS CORPORATION
SAN DIEGO

CARD READ AND/OR PUNCH ROUTINE

This routine combines the features of the card read IC 005 and the card punch IC 006 routines in order that the operations of punching and reading may be performed simultaneously. This routine requires 401 octal words of ES in which to operate, constants and temporary storages included. A two cycle operating basis, which more efficiently uses the card cycle time, makes possible the reading of as many as thirty fields from a card and the punching of as many as thirty fields in a card simultaneously and have 14 ms computing time available between references to the routine. If the routine is used for reading alone or for punching alone, these cards may contain as many as forty fields.

During the first five points of the 18 point card cycle, information to be punched is converted to decimal and stored in card code in a card image; final conversion and scaling of the information read from the previous card is performed and stored in specified ES memory locations. The remainder of the card cycle is used for punching and reading, row by row with the following 3 steps occurring at each row of the card cycle (beginning at row 9 and continuing through row 12):

1. Punch information ⟶ card machine.
2. Read information ⟶ 1103.
3. Read information converted to B. C. D. and stored for final conversion during next card cycle.

Although two card cycles are necessary to complete the operations, the net effect is the converting of one read card and the punching of one card during each card cycle.

The card routine requires the following information:

1) Binary scaling.

2) Decimal scaling.

3) Location of fields on the cards.

4) Zero suppression (if punching).

This information is supplied the routine in a standard form called a parameter word. One parameter word is required for each card field whether reading or punching. If both reading and punching are to be accomplished, a parameter word is required for each.

A field consists of a number of consecutive card columns. The last column of a field is reserved for the sign of the decimal number stored in the field. An 11-punch signifies a negative number, no punch (blank column) signifies a positive number. A combination 12, 3 and 8 punch in one column signifies a decimal point.

Fields need not be adjacent--there may be unused columns punched or unpunched between them--nor need they be alike in size.

The conversion operations are automatic.

Enty into the card read and punch routine from line y is effected as follows:

y)	37	Ommmm	Ommmm	To read and punch routine
y + 1)	AB	$OPPPP_1$	$ODDDD_1$	Read control word
y + 2)	00	$OPPPP_2$	$ODDDD_2$	Punch control word
y + 3)	NI	uuuuu	vvvvv	Next instruction

Ommmm represents the beginning ES operating address of the read and punch routine.

PX 71900-9-(129)

The 37 command records in Ommmm the address of the control word. The routine is then entered at Ommmm and after finishing the operations the card routine exits to y + 3, the line following the second control word.

CONTROL WORD

Since this routine is coded to read and punch with each reference to the routine, it is necessary to use two control words with each reference to it. Their composition is explained below.

AB $OPPPP_1$ $ODDDD_1$

00 $OPPPP_2$ $ODDDD_2$

A, the first octal digit, controls positioning of cards in the read and punch channels of the Bull Reproducer.

B, the second octal digit, controls the operation to be performed.

AB

00 cycle Bull Reproducer. Cards already in either channel are advanced one card station. This is also performed with any of the following operations.

01 (not allowed)

02 punch a card

10 pick a card from the read hopper. This card will not go into the read channel until the next read card is picked.

11 pick a card from the read hopper and read a card.

12 pick a card from the read hopper and punch.

20 pick a card from the punch hopper. This card will not go into the punch channel until another card is picked from the punch hopper.

21 not allowed.

22 pick a card from the punch hopper and punch a card.

30 pick a card from both hoppers.

31 pick a card from both hoppers and read a card.

32 pick a card from both hoppers and punch a card.

33 pick a card from both hoppers and both read a card and punch a card.

00, the first two octal digits of the second control word always remain zeros.

$OPPPP_1$ is the ES address of the first parameter word for the card read portion.

$ODDDD_1$ is the E^S storage address for the first word read from the read card.

$OPPPP_2$ is the ES address of the first parameter word for the card punch portion.

$ODDDD_2$ is the ES address of the first data word for the card punch portion.

PARAMETER WORDS

Parameter words consist of twelve octal digits divided into six groups of two each.

FF SS BB LL RR ZZ

FF: Flag for final parameter word.

FF = 77 octal for final word.

FF = 00 otherwise.

SS: Binary scaling factor. (number of bits to the right of the binary point.)

BB: Number of blanks or unused columns between previous field, or edge of card, and present field.

LL: Number of digit positions to the left of the decimal point.

RR: Number of remaining columns in the field exclusive of sign.

RR = 00 no decimal point and no decimal fraction.

ZZ: Flag for zero suppression.

ZZ = 77 octal for zero suppression.

ZZ = 00 for no zero suppression. These two digits are decoded by the punch routine only. Only zeros in the integer part are suppressed. A zero immediately preceding the decimal point is not suppressed.

Total size of a field = LL + RR + 1.

RANGE OF PARAMETERS:

DECIMAL	OCTAL
$00 \leq SS \leq 35$	$00 \leq SS \leq 43$
$00 \leq BB \leq 63$	$00 \leq BB \leq 77$
$00 \leq LL \leq 10$	$00 \leq LL \leq 12$
$00 \leq RR \leq 11$	$00 \leq RR \leq 13$
$01 \leq LL + RR \leq 11$	$01 \leq LL + RR \leq 13$

The parameter words, one for each field, must be stored consecutively starting at some ES memory location OPPPP. There must also be an equal number of consecutive words starting with some ES memory location ODDDD.

Reading occurs at the second card station in the read channel and punching occurs at the third card station in the punch channel. Therefore, these cards must be advanced to these respective positions before these operations can take place. This may be done manually or the routine may be used to position the cards as described earlier.

It should be noted that once a card has entered either the read or punch channel it continues to advance one card station each time the Bull Reproducer is cycled.

The routine may be programmed in the following manner:

READ ONLY:

37	01000	01000	
10	00000	00000	pick a read card.
00	00000	00000	
37	01000	01000	
11	00000	00000	pick a read card and read a card.
00	00000	00000	
37	01000	01000	pick a read card, read a card
11	$0PPPP_1$	$0DDDD_1$	and convert.
00	00000	00000	

PUNCH ONLY:

37	01000	01000	
20	00000	00000	pick a punch card.
00	00000	00000	
37	01000	01000	
20	00000	00000	pick a punch card.
00	00000	00000	
37	01000	01000	
22	00000	00000	pick a punch card, and punch a card.
00	$0PPPP_2$	$0DDDD_2$	

PX 71900-9-(129)

READ and PUNCH:

37	01000	01000	
30	00000	00000	pick from both hoppers
00	00000	00000	
37	01000	01000	
31	00000	00000	pick from both hoppers and read a
00	00000	00000	card.
37	01000	01000	pick from both hoppers convert, punch
33	$OPPPP_1$	$ODDDD_1$	and read.
00	$OPPPP_2$	$ODDDD_2$	

Numbers are rounded to the desired number of decimal digits before punching takes place. A divide check error stop results if an insufficient number of card columns is allowed for the integer portion of a field.

In case of a card machine failure or an accidental stop in the middle of a card cycle, the current card may be reread or punched again; reposition the cards set (P A K) = 00000, and START.

This routine is coded in standard form. All constants are contained by the routine.

Number of words:

Used by the routine: $(322)_8 = (210)_{10}$

Used by temporaries immediately following routine: $(57)_8 = (47)_{10}$

For assembly modification: $(271)_8 = (185)_{10}$

Used for constants: $(31)_8 = (25)_{10}$

Thirty fields may be read and thirty fields may be punched simultaneously.

Forty fields may be read or forty fields may be punched.

14 ms computing time is available between references to the card routine.

CARD READ AND PUNCH RT. IC007

1000	71 01276 00000		CONTROL WORD 1 → (A)
1001	15 20000 01034		STORE P_1
1002	16 20000 01072		STORE D_1
1003	16 01124 00000		SET EMERGENCY RERUN
1004	55 20000 00003		SHIFT FIRST DIGIT → (Q)
1005	31 01273 00003		BULL CODE · 2^{-2} → (A)
1006	52 01312 20000		EXTRACT PICK CODES
1007	32 20000 00001		SL_2 (A_R)
1010	44 01011 01011		SL_1 (A_L)
1011	44 01133 01012		PUNCH ?
1012	16 01261 01120		NO: SET α_3
1013	44 01017 01014		READ ?
1014	37 01160 01121		NO: SET γ_2
1015	41 01107 01152		ALL ROWS PUNCHED ?
1016	56 00000 01016		EXIT
1017	37 01160 01024		SET γ_1
1020	76 00000 01376		READ
1021	76 10000 01346		ONE
1022	76 10000 01362		ROW
1023	45 00000 31023		A-SWITCH
1024	32 01276 00000		ADD READ CODE
1025	37 01023 01121		SET A_1
1026	41 01127 01076		DIGIT-1 → DIGIT, NEG. ?
1027	16 01101 01127		SIGN SENTINEL → DIGIT
1030	37 01023 01076		SET A_2
1031	15 01263 01253		PRESTORE MATRIX TRANSFER
1032	37 01122 01016		SET β_2
1033	37 01075 01034		SET CONVERSION REPEAT
1034	55 30000 00000		PARAMETER WORD 1 → (Q)
1035	44 01036 01037		LAST FIELD ?
1036	16 01122 01075		YES: SET CONVERSION EXIT
1037	55 10000 00013		SL_{11} (Q)

CARD READ AND PUNCH

1040	51	01313 20000	S → (A)
1041	16	20000 01063	SET SHIFT
1042	55	10000 00006	STORE
1043	51	01313 01233	B
1044	55	10000 00006	STORE
1045	51	01313 01227	L
1046	55	10000 00006	STORE
1047	51	01313 01107	R
1050	32	20000 00016	$R \cdot 2^{15} \rightarrow (A)$
1051	35	01264 01052	SET NEXT INSTRUCTION
1052	00	77777 00000	STORE 10^{R-1}
1053	37	01053 01054	SWITCH
1054	41	01233 01247	SHIFT B DIGITS
1055	37	01260 01056	CONVERT
1056	41	01227 01250	L TERMS
1057	37	01254 01061	SKIP DECIMAL POINT
1060	37	01260 01061	CONVERT
1061	41	01107 01250	REMAINING TERMS
1062	54	01052 10107	$1/2\ 10^{R-1} \rightarrow$ ROUNDING TERMS
1063	31	01233 30000	$\|N\| \cdot 10^{R-1} \cdot 2^{S} \rightarrow (A)$
1064	32	10000 00000	ADD ROUNDING TERM
1065	73	01052 01226	STORE $\|N\| \cdot 2^{S}$
1066	37	01254 01250	SHIFT SIGN
1067	51	01271 20000	SIGN SENTINEL → (A)
1070	47	01071 01072	IS N NEGATIVE ?
1071	13	01226 01226	YES: $-\|N\| \cdot 2^{S} \rightarrow N \cdot 2^{S}$
1072	11	01226 30000	STORE RESULT
1073	21	01034 01314	ADVANCE P
1074	21	01072 01276	ADVANCE D
1075	45	00000 31075	
1076	11	01263 01107	PRESET MATRIX STORE
1077	55	01346 00000	ROW WORD 1 → (0)

PX 71900-9-(129)

CARD READ AND PUNCH

```
1100  11 01312 01346    SET INDEX = 3
1101  31 01275 00011    SENTINEL ──►(A_L)
1102  32 20000 00003    SL_4 (A_R) • SL_3 (A_L)
1103  44 01104 01105    BIT THIS COL. ?
1104  32 01127 00000     YES: ADD DIGIT
1105  46 01106 01102    SENTINEL REACHED ?
1106  31 20000 00000     YES: CLEAR (A_L)
1107  00 00000 00000    ADD TO MATRIX
1110  21 01107 01315     STEP MATRIX STORE INSTRUCTIONS
1111  41 01346 01101    ROW WORD USED UP ?
1112  37 01112 01113    SWITCH
1113  55 01362 00000    ROW WORD 2 ──►(Q)
1114  37 01112 01100    STORE BCD INFO. ──►MATRIX
1115  55 01376 00034    ROW WORD 3
1116  37 01112 01101    BCD INFO. ──►MATRIX
1117  16 01124 00000    SET EMERGENCY RERUN
1120  45 00000 31120    ∝ -SWITCH
1121  17 00000 20000    START BULL
1122  37 01122 01123    READ CONVERSION SWITCH β
1123  36 20000 01107    CLEAR PUNCH INDEX
1124  27 20000 01000    SET TO PICKUP
1125  35 01276 01127       NEXT CONTROL WORD
1126  35 01275 01016    SET EXIT
1127  00 00000 00000    CONTROL WORD ──►(A)
1130  11 01300 01127    10 ──►DIGIT
1131  75 10055 01117    CLEAR MATRIX AND
1132  11 01107 01322       CARD IMAGE
1133  32 01273 00000    ADD PUNCH CODE
1134  37 01120 01013    SET ∝_2
1135  15 20000 01143    STORE P_2
1136  32 20000 00016    SL_15 (A)
1137  15 20000 01177    STORE D_2
```

CARD READ AND PUNCH

1140	11 01272 01377		PRESTORE COLUMN SELECTOR
1141	11 01265 01400		PRESTORE ROW SELECTOR
1142	37 01155 01143		SET SWITCH
1143	55 30000 00000		PARAMETER WORD $\longrightarrow$(Q)
1144	44 01145 01161		LAST FIELD ?
1145	37 01155 01161		SET SWITCH
1146	16 01161 01107		SET PUNCH INDEX
1147	75 30003 01151		SET UP
1150	16 01020 01155		PUNCH ORDERS
1151	37 01120 01154		SET α_1
1152	75 20003 01154		STEP
1153	23 01155 01276		PUNCH ORDERS
1154	16 01124 00000		SET EMERGENCY RERUN
1155	77 00000 30000		PUNCH
1156	77 10000 30000		ONE
1157	77 10000 30000		ROW
1160	45 00000 31160		τ -SWITCH
1161	55 10000 00013		SL_{11} (Q)
1162	51 01313 20000		S $\longrightarrow$(A)
1163	33 20000 00000		-S $\longrightarrow$(A)
1164	35 01267 01200		SET SHIFT
1165	37 01053 01042		STORE B•L•R• 10^{R-1}
1166	44 01220 01167		ZERO SUPPRESS ?
1167	16 01231 01246		SET FOR NO ZERO SUPPRESS
1170	37 01241 01171		SHIFT
1171	41 01233 01234		B COLUMNS
1172	31 01227 00017		L • 2^{15} $\longrightarrow$(A)
1173	35 01270 01174		SET NEXT INSTRUCTION
1174	00 00000 00000		STORE 10^L
1175	31 01314 00023		2^{34} $\longrightarrow$(A)
1176	73 01052 10000		$1/2 \cdot 2^{35}/10^{R-1}$ $\longrightarrow$ROUNDING TERM
1177	12 30000 01233		STORE $\lvert N \rvert \cdot 2^S$

CARD READ AND PUNCH

1200	00 00000 00000	$N \cdot 2^{35} \rightarrow (A)$
1201	32 10000 00000	ADD ROUNDING TERM
1202	73 01174 20000	$\|N\| \cdot 2^{35}/10^{L} \rightarrow (A)$
1203	35 20000 01174	STORE $\|N\| \cdot 2^{36}/10^{L}$
1204	37 01241 01205	L TIMES
1205	41 01227 01242	THRU CONVERSION
1206	16 01262 01241	DECIMAL POINT
1207	41 01107 01223	AND REMAINING TERMS
1210	41 01107 01242	$\rightarrow$IMAGE
1211	15 01177 01216	SET TO CHECK SIGN
1212	21 01143 01314	STEP P
1213	21 01177 01314	STEP D
1214	16 01155 01241	SET EXIT
1215	33 01314 00000	$-1 \rightarrow (A)$
1216	55 30000 00000	$N \cdot 2^{8} \rightarrow (Q)$
1217	44 01232 01234	NEGATIVE ?
1220	37 01246 01170	SET FOR ZERO SUPPRESS
1221	43 01227 01231	NO ZERO SUPPRESS IF L = 0
1222	45 00000 01234	
1223	31 01321 00000	$3 \rightarrow (A)$
1224	35 01400 01226	SET FOR 3 PUNCH
1225	35 01274 01227	SET FOR 8 PUNCH
1226	00 00000 00000	STORE 3 PUNCH
1227	00 00000 00000	STORE 8 PUNCH
1230	33 01320 00000	SET FOR 12 PUNCH
1231	37 01246 01232	SET NO ZERO SUPPRESS
1232	35 01400 01233	SET NEXT INSTRUCTION
1233	00 00000 00000	STORE DIGIT IN IMAGE
1234	55 01377 00043	SR_1 COLUMN SELECTOR
1235	44 01236 01241	ADVANCE TO NEXT CARD FIELD ?
1236	21 01400 01317	YES: STEP ROW SELECTOR 12 ROWS
1237	42 01266 01241	THIRD FIELD ?

PX 71900-9-(129)

CARD READ AND PUNCH

1240	55	01377	00010	YES: USE LAST 8 COLUMNS
1241	45	00000	31241	
1242	31	01174	00002	4N → (A)
1243	32	01174	00001	10N → (A)
1244	11	20000	01174	STORE FRACTIONAL PART
1245	34	20000	00063	INTEGER PART → (R)
1246	47	01231	31246	DIGIT = 0 ?
1247	16	01053	01254	
1250	41	01127	01253	MATRIX INDEX
1251	21	01253	01314	RESET INDEX
1252	16	01240	01127	TRANSFER MATRIX WORD
1253	55	30000	00004	SHIFT TO NEXT DECIMAL DIGIT
1254	37	01254	01255	SWITCH
1255	31	01233	00002	4N → (A)
1256	32	01233	00001	10N → (A)
1257	52	01316	01233	10N PLUS DIGIT → N
1260	45	00000	31260	
1261	00	00000	01160	
1262	00	00000	01210	
1263	35	01322	01322	PRESET
1264	11	01276	01052	PRESET
1265	27	01335	01377	PRESET
1266	27	01363	01377	PRESET
1267	31	01233	00043	PRESET
1270	11	01277	01174	PRESET
1271	00	00000	00014	12
1272	40	00000	00000	PRESET FOR COLUMN SELECTOR
1273	00	00000	00002	2
1274	00	00005	00000	$5 \cdot 2^{15}$
1275	54	00000	00000	MODIFY COMMAND
1276	00	00000	00001	1
1277	00	00000	00001	TABLE

PX 71900-9-(129)

CARD READ AND PUNCH

1300	00 00000 00012	
1301	00 00000 00144	POWERS
1302	00 00000 01750	
1303	00 00000 23420	OF
1304	00 00003 03240	
1305	00 00036 41100	TEN
1306	00 00461 13200	
1307	00 05753 60400	
1310	00 73465 45000	
1311	11 24027 62000	
1312	00 00000 00003	3
1313	00 00000 00077	EXTRACTOR
1314	00 00001 00000	U ADVANCE
1315	00 00001 00001	U AND V ADVANCE
1316	00 00000 00017	4 BIT EXTRACTOR
1317	00 00014 00000	U ADVANCE
1320	00 00002 00000	2^{15}
1321	00 00003 00000	$3 \cdot 2^{15}$
1322	00 00000 00000	MATRIX WORD 1
1323	00 00000 00000	2
1324	00 00000 00000	3
1325	00 00000 00000	4
1326	00 00000 00000	5
1327	00 00000 00000	6
1330	00 00000 00000	7
1331	00 00000 00000	8
1332	00 00000 00000	9
1333	00 00000 00000	IMAGE FIELD I ROW 12
1334	00 00000 00000	11
1335	00 00000 00000	0
1336	00 00000 00000	1
1337	00 00000 00000	2

PX 71900-9-(129)

CARD READ AND PUNCH

1340	00 00000 00000		3
1341	00 00000 00000		4
1342	00 00000 00000		5
1343	00 00000 00000		6
1344	00 00000 00000		7
1345	00 00000 00000		8
1346	00 00000 00000		9
1347	00 00000 00000	IMAGE FIELD II ROW	12
1350	00 00000 00000		11
1351	00 00000 00000		0
1352	00 00000 00000		1
1353	00 00000 00000		2
1354	00 00000 00000		3
1355	00 00000 00000		4
1356	00 00000 00000		5
1357	00 00000 00000		6
1360	00 00000 00000		7
1361	00 00000 00000		8
1362	00 00000 00000		9
1363	00 00000 00000	IMAGE FIELD III ROW	12
1364	00 00000 00000		11
1365	00 00000 00000		0
1366	00 00000 00000		1
1367	00 00000 00000		2
1370	00 00000 00000		3
1371	00 00000 00000		4
1372	00 00000 00000		5
1373	00 00000 00000		6
1374	00 00000 00000		7
1375	00 00000 00000		8
1376	00 00000 00000		9
1377	00 00000 00000	COLUMN SELECTOR	
1400	00 00000 00000	ROW SELECTOR	

CARD PUNCH ROUTINE IC006

This routine converts specified binary numbers into decimal and causes them to be punched into cards. It requires 245 octal words of ES in which to operate, constants and temporary storage included. It is possible to convert and punch as many as forty fields in a card and have 17 ms computing time available between references to the punch routine.

The following information is required:

1) Binary scaling

2) Decimal scaling

3) Locations of fields on the card

4) Zero suppression

This information is supplied to the card routine in a standard form called a parameter word. One parameter word is required for each field.

A field consists of a number of consecutive card columns. The last column of the field is reserved for the sign of the decimal number stored in that field. An 11-punch signifies a negative number, no punch (blank column) signifies a positive number. A combination 12, 3 and 8 punch in one column represents a decimal point.

Fields need not be adjacent--there may be unused columns, punched or unpunched, between them--nor need they be alike in size.

This routine is entered from line y as follows:

y) 37 Onnnnn Onnnnn (to card routine)

y + 1) AB OPPPP ODDDD (control word)

y + 2) Next instruction

Onnnnn represents the beginning address (ES operating address) of the card routine.

The 37 command records in Ommmm the address of the control word. The routine is then entered at Ommmm. After finishing its operation, the card routine exits to y + 2, the line following the control word.

CONTROL WORD

The control word controls the operation of the card routine Its composition is as follows:

AB OPPPP ODDDD

A. The first octal digit, controls positioning of cards in the read and punch channels of the Bull Reproducer.

A = 1 Pick a card from the read hopper.

A = 2 Pick a card from the punch hopper.

B. The second octal digit, controls the operation to be performed.

B = 2 punch a card.

OPPPP is the address of the first parameter word.

ODDDD is the address of the first data word.

OPPPP and ODDDD must both be ES addresses.

PARAMETER WORD

A parameter word consists of twelve octal digits divided into six groups of two each:

FF SS BB LL RR ZZ

FF: Flag for final parameter word.

FF = 77 octal for final word.

FF = 00 otherwise.

SS: Binary scaling factor. (Number of bits to the right of the binary point.)

BB: Number of blank or unused columns between previous field, or edge of card, and present field.

LL: Number of digit positions to the left of the decimal point.

RR: Number of remaining columns in the field exclusive of sign.

RR = 00 no decimal point and no decimal fraction.

ZZ: Flag for zero suppression.

ZZ = 77 octal for zero suppression.

ZZ = 00 for no zero suppression. Only zeros in the integer part are suppressed. A zero immediately preceding the decimal point is not suppressed.

Total size of a field LL + RR + 1

Range of Parameters:

DECIMAL	OCTAL
$00 \leq SS \leq 35$	$00 \leq SS \leq 43$
$00 \leq BB \leq 63$	$00 \leq BB \leq 77$
$00 \leq LL \leq 10$	$00 \leq LL \leq 12$
$00 \leq RR \leq 11$	$00 \leq RR \leq 13$
$01 \leq LL + RR \leq 11$	$01 \leq LL + RR \leq 13$

The parameter words, one for each field, must be stored consecutively starting at some ES memory location OPPPP. There must be an equal number of consecutive words starting with some ES memory location ODDDD, filled with data for the punch routine.

Punching takes place at the third card station in the punch channel, therefore, two punch cards must be advanced before punching can take place. This can be done manually, or the punch routine can be used to position the cards as follows:

37	Omnnnn	Omnnnn	(to card routine)
20	00000	00000	(pick punch card)
37	Omnnnn	Omnnnn	(to card routine)
20	00000	00000	(pick punch card)

It should be noted that once a card has entered either the read or punch channel it continues to advance one card station each time the Bull Reproducer is cycled.

Numbers are rounded to the specified number of decimal digits after the decimal point before punching takes place. A divide check error stop results if an insufficient number of card columns is allowed for the integer portion of a field.

In the event of a card machine failure or an accidental stop in the middle of a card cycle, the current card may be punched again: Set (P A K) = 00000, and START.

The routine is coded in standard form; all constants are contained by the routine.

Number of cells used by the routine: $(201)_8 = (129)_{10}$

Number of temporaries immediately following routine: $(44)_8 = (36)_{10}$

Number of words for assembly modification: $(153)_8 = (107)_{10}$

Number of constants: $(26)_8 = (22)_{10}$

$(40)_{10}$ fields may be punched in a card.

17 ms computing time available between references to punch routine.

CARD PUNCH ROUTINE IC006

```
1000  71 01160 30000      CONTROL WORD ⟶(A)
1001  15 20000 01027      SET PARAMETER PICKUP
1002  55 20000 00003      CONTROL WORD • 2^3 ⟶(A) • (Q)
1003  32 20000 00013      CONTROL WORD • 2^15 ⟶(A)
1004  15 20000 01057      SET DATA PICKUP
1005  31 01155 00003      BASIC BULL CODE ⟶(A)
1006  52 01174 20000      EXTRACT PICK CODES
1007  32 20000 00001      SL2  (A)
1010  44 01011 01011      SL1  (Q)
1011  44 01016 01012      PUNCH ?
1012  17 00000 20000      START BULL
1013  31 01000 00000      SET
1014  35 01157 01015         EXIT
1015  56 00000 01015      EXIT
1016  32 01155 00000      ADD PUNCH CODE
1017  17 00000 20000      START BULL
1020  27 01201 20000      CLEAR
1021  75 10043 01023
1022  11 20000 01202        IMAGE
1023  16 01152 00000      SET EMERGENCY RERUN
1024  11 01154 01015      PRESTORE COLUMN SELECTOR
1025  11 01141 01110      PRESTORE ROW SELECTOR
1026  37 01112 01027      SET SWITCH
1027  55 30000 00000      PARAMETER WORD ⟶(Q)
1030  44 01103 01031      LAST FIELD ?
1031  55 10000 00013      SL11  (Q)
1032  51 01176 20000      S ⟶ A
1033  33 20000 00000      -S ⟶ A
1034  35 01146 01060      SET UP SHIFT ORDER
1035  55 10000 00006      STORE
1036  51 01175 01120        B
1037  55 10000 00006      STORE
```

CARD PUNCH ROUTINE

1040	51	01175	01111	L
1041	32	20000	00016	$L \cdot 2^{15} \rightarrow (A)$
1042	35	01147	01043	SET NEXT INSTRUCTION
1043	00	00000	00000	STORE 10^L
1044	55	10000	00006	STORE
1045	51	01175	01043	R
1046	32	20000	00016	$R \cdot 2^{15} \rightarrow (A)$
1047	35	01150	01050	SET NEXT INSTRUCTION
1050	00	00000	00000	STORE 10^{R-1}
1051	44	01100	01052	ZERO SUPPRESSION ?
1052	16	01123	01140	SET NO ZERO SUPPRESSION
1053	37	01133	01054	SHIFT P COLUMNS
1054	41	01120	01126	
1055	31	01176	00023	$2^{34} \rightarrow (A)$
1056	73	01050	10000	$1/2\ 10^{-(R-1)} \cdot 2^{35} \rightarrow$ ROUNDING TERM
1057	12	30000	01120	STORE $\lvert N \rvert \cdot 2^{36}$
1060	00	00000	00000	$\lvert N \rvert \cdot 2^{35} \rightarrow (A)$
1061	32	10000	00000	ADD ROUNDING TERM
1062	73	01121	20000	$N \cdot 2^{35} / 10^L \rightarrow (A)$
1063	35	20000	01050	STORE $\lvert N \rvert \cdot 2^{36} / 10^L$
1064	37	01133	01065	L TIMES THRU
1065	41	01111	01124	CONVERSION LOOP
1066	16	01151	01133	STORE DECIMAL POINT
1067	41	01043	01115	REMAINING
1070	41	01043	01134	TERMS
1071	15	01057	01076	SET TO CHECK SIGN OF N
1072	21	01027	01176	STEP PARAMETER
1073	21	01057	01176	STEP D
1074	16	01112	01133	SET EXIT
1075	33	01176	00000	$-1 \rightarrow (A)$
1076	55	30000	00000	
1077	44	01124	01126	NEGATIVE ?

PX 71900-9-(130)

CARD PUNCH ROUTINE

```
1100  37 01140 01053      SET FOR ZERO SUPPRESSION
1101  43 01111 01123      NO ZERO SUPPRESSION, IF L = 0
1102  45 00000 01126
1103  37 01112 01031      SET SWITCH
1104  75 30003 01107      SET UP
1105  11 01143 01110        PUNCH ORDERS
1106  43 01144 01013      ALL 12 ROWS PUNCHED ?
1107  16 01152 00000      SET EMERGENCY RERUN
1110  00 00000 00000      PUNCH
1111  00 00000 00000            ONE
1112  00 00000 00000               ROW
1113  75 20003 01106      STEP
1114  23 01110 01160         PUNCH ORDERS
1115  31 01177 00000      3 ⟶ (A)
1116  35 01110 01120      SET FOR 3 PUNCH
1117  35 01156 01121      SET FOR 8 PUNCH
1120  00 00000 00000      STORE 3 PUNCH
1121  00 00000 00000      STORE 8 PUNCH
1122  33 01200 00000      SET FOR 12 PUNCH
1123  37 01140 01124      SET NO ZERO SUPPRESSION
1124  35 01110 01125      SET NEXT INSTRUCTION
1125  00 00000 00000      STORE DIGIT PUNCH
1126  55 01015 00043      SR1 COLUMN SELECTOR
1127  44 01130 01133      ADVANCE TO NEXT CARD FIELD ?
1130  21 01110 01153      STEP ROW SELECTOR 12 ROWS
1131  42 01142 01133      THIRD CARD FIELD ?
1132  55 01015 00010      USE LAST 8 COLUMNS
1133  45 00000 31133
1134  31 01050 00002      4N ⟶ (A)
1135  32 01050 00001      10N ⟶ (A)
1136  11 20000 01050      STORE FRACTIONAL PART
1137  34 20000 00063      INTEGER PART ⟶ (A)
```

PX 71900-9-(130)

CARD PUNCH ROUTINE

1140	47 01123 31140	DIGIT = 0 ?
1141	27 01203 01015	PRESET
1142	27 01231 01015	PRESET
1143	77 00000 01244	PRESET
1144	77 10000 01214	PRESET
1145	77 10000 01230	PRESET
1146	31 01120 00043	PRESET
1147	11 01161 01121	PRESET
1150	11 01160 01060	PRESET
1151	00 00000 01070	
1152	00 00000 01000	
1153	00 00014 00000	U ADVANCE
1154	40 00000 00000	PRESET FOR COLUMN SELECTOR
1155	00 00000 00002	2
1158	00 00005 00000	$5 \cdot 2^{15}$
1157	54 00000 00000	$54 \cdot 2^{30}$
1160	00 00000 00001	1
1161	00 00000 00001	TABLE
1162	00 00000 00012	
1163	00 00000 00144	POWERS
1164	00 00000 01750	
1165	00 00000 23420	OF
1166	00 00003 03240	
1167	00 00036 41100	TEN
1170	00 00461 13200	
1171	00 05753 60400	
1172	00 73465 45000	
1173	11 24027 62000	
1174	00 00000 00003	3
1175	00 00000 00077	EXTRACTOR
1176	00 00001 00000	$1 \cdot 2^{15}$
1177	00 00003 00000	$3 \cdot 2^{15}$

CARD PUNCH ROUTINE

1200	00 00002 00000	2^{15}
1201	00 00000 00000	CARD FIELD I ROW 12
1202	00 00000 00000	11
1203	00 00000 00000	0
1204	00 00000 00000	1
1205	00 00000 00000	2
1206	00 00000 00000	3
1207	00 00000 00000	4
1210	00 00000 00000	5
1211	00 00000 00000	6
1212	00 00000 00000	7
1213	00 00000 00000	8
1214	00 00000 00000	9
1215	00 00000 00000	CARD FIELD II ROW 12
1216	00 00000 00000	11
1217	00 00000 00000	0
1220	00 00000 00000	1
1221	00 00000 00000	2
1222	00 00000 00000	3
1223	00 00000 00000	4
1224	00 00000 00000	5
1225	00 00000 00000	6
1226	00 00000 00000	7
1227	00 00000 00000	8
1230	00 00000 00000	9
1231	00 00000 00000	CARD FIELD III ROW 12
1232	00 00000 00000	11
1233	00 00000 00000	0
1234	00 00000 00000	1
1235	00 00000 00000	2
1236	00 00000 00000	3
1237	00 00000 00000	4

CARD PUNCH ROUTINE

1240	00 00000 00000	5
1241	00 00000 00000	6
1242	00 00000 00000	7
1243	00 00000 00000	8
1244	00 00000 00000	9

TWO CYCLE READ ONLY CARD ROUTINE IC005

This new card routine operates on a two cycle basis, making more efficient use of the card cycle time than the previous routine. As a result, as many as forty fields may be read from each card without causing any timing difficulties. In addition, this new routine requires less ES space than the former read routine.

Two basic operations are performed by this routine during the 18-point card cycle. The first five points (about 140 ms) are used to decode the control word and to perform the final conversion of information read during the previous read cycle. The remainder of the card cycle is used to read information from the present card and convert this information into binary coded decimal form. The binary coded decimal information is then converted to binary and scaled during the first part of the next card cycle. Thus, although it takes two card cycles to complete the operation of reading and converting, the net effect is conversion of one card each card cycle.

A time of as much as 14 ms may be used for computation between references to the read routine without causing the bull to skip a cycle.

The read routine may be used to perform any combination of the following operations according to the contents of a control word:

1. Pick (Prime) a read card
2. Pick (Prime) a punch card
3. Pick a read card and read

The conversion operation is automatic and is always performed during the card cycle occurring with the next use of the read routine. This card routine requires the following information:

1. Binary scaling
2. Decimal scaling
3. Locations of fields on the card.

This information is supplied to the card routine in a standard form called a parameter word. One parameter word is required for each card field.

A field consists of a number of consecutive card columns. The last column of a field is reserved for the sign of the decimal number stored in that field. An 11-punch signifies a negative number, no punch (blank column) signifies a positive number. A combination 12, 3 and 8 punch in one column may be used to represent a decimal point.

Fields need not be adjacent--there may be unused columns, punched or unpunched, between them--nor need they be alike in size.

Entry to the card read routine from line y is effected as follows:

(y)	37	Ommmm	Ommmm	(To read routine)
(y+1)	AB	OPPPP	ODDDD	(Control word)
(y+2)	NI	uuuuu	vvvvv	(next instruction)

Ommmm represents the beginning address (ES operating address) of the card routine. The control word is described below.

The 37 command records in Ommmm the address of the control word. The routine is then entered at Ommmm. After finishing its operation, the card routine exits to y+2, the line following the control word.

Control Word Composition:

A: First octal digit, controls picking of cards in either channel of the bull reproducer.

A=0 Do not pick
A=1 Pick read card
A=2 Pick punch card
A=3 Pick both read and punch

B: Second octal digit, controls the reading operation.

B=0 Do not read
B=1 Read a card

OPPPP ES Address of the first parameter word.

ODDDD ES Address where the number from the first card field is to be stored.

OPPPP and ODDDD are required only if conversion is being performed during the card cycle.

One parameter word is required for each card field. Parameter words must be stored consecutively beginning at address OPPPP. Numbers read from the card are stored consecutively beginning at address ODDDD.

Parameter Word Composition:

FF SS BB LL RR ZZ

FF: Flag for final parameter word
FF=77 (octal) for final parameter word
FF=00 otherwise

SS: Binary scaling factor (number of bits to the right of the binary point) of converted number.

BB: Number of blank or unused card columns to the left of the field.

LL: Number of columns (digit positions) to the left of the decimal point.

RR: Number of remaining columns in the field, exclusive of sign (number of decimal digits to the right of the decimal point plus one for the decimal point).

RR=00 indicates no decimal fraction and no decimal point.

ZZ: Not used.

Range of Parameters:

Decimal	Octal
$00 \leq SS \leq 35$	$00 \leq SS \leq 43$
$00 \leq BB \leq 63$	$00 \leq BB \leq 77$
$00 \leq LL \leq 10$	$00 \leq LL \leq 12$
$00 \leq RR \leq 11$	$00 \leq RR \leq 13$
$01 \leq LL + RR \leq 11$	$01 \leq LL + RR \leq 13$

Reading takes place at the second card station in the read channel--one read card must be advanced before reading takes place. This may be done manually, or may be done as follows:

37	0[illegible]	0[illegible]	(to card routine)
10	00000	00000	(pick read card)

The card just advanced will not feed further unless another order to pick a card is given--both pick and read orders must be given to read this card.

It should be noted that once a card enters either the read or punch channel it continues to advance one card station each time the Bull Reproducer is cycled.

The information read from the card is stored within the card routine in coded decimal form. Thus, if the subroutine is destroyed between card cycles, this information will not be converted on the following card cycle.

Use of the read routine causes the Bull Reproducer to go through one card cycle; if a series of cards is to be read, there must be a reference to the read routine for each card.

Example of coding used to read a stack of n cards:

```
37   Cxxxxx   Cxxxxx
10   00000    00000    Pick card #1

37   Cxxxxx   Cxxxxx
11   00000    00000    Pick card 2, read card 1

37   Cxxxxx   Cxxxxx
11   OPPPP    ODDDD    Pick card 3, read card 2, conv. card 1

37   Cxxxxx   Cxxxxx
11   OPPPP    ODDDD    Pick card 4, read card 3, conv. card 2
 .     .        .
 .     .        .
 .     .        .
37   Cxxxxx   Cxxxxx
11   OPPPP    ODDDD    Pick card n+1, read card n, conv. card n-1

37   Cxxxxx   Cxxxxx
00   OPPPP    ODDDD    Convert card n
```

In case of a card machine failure or an accidental stop in the middle of a card cycle, the current card may be reread: reposition the cards, set (PAK)=00000, and start 1103.

This routine is coded in standard form.

All constants are contained by the routine.

Number of words:

Used by the routine: $(164)_8 = (116)_{10}$
Used by temporaries: $(14)_8 = (12)_{10}$
For assembly modification: $(141)_8 = (97)_{10}$
Used for constants: $(23)_8 = (19)_{10}$

Forty fields may be read from a card.

14ms computing time available between references to the card routine.

PX 71900-9-(131)

CARD READ ROUTINE IC005

```
1000  71 01142 30000    CONTROL WORD ⟶A
1001  15 20000 01061    STORE PARAMETER WORD
1002  16 20000 01116    STORE DATA WORD
1003  55 20000 00003    CONTROL WORD ⟶(Q)
1004  31 01163 00001    BULL CODE · 2^-2 ⟶(A)
1005  52 01156 20000    EXTRACT PICK CODES
1006  32 20000 00001    SL2   (A)
1007  55 10000 00002    SL1   (Q)
1010  44 01017 01011    READ ?
1011  17 00000 20000    START CARD CYCLE
1012  37 01012 01013    CONVERSION SWITCH
1013  37 01013 01014    READ SWITCH
1014  31 01000 00000    SET
1015  35 01141 01016      EXIT
1016  00 00000 00000    ROW WORD 3
1017  32 01142 00000    ADD READ CODE
1020  37 01013 01011    SET TO READ
1021  36 20000 01176    STORE 0
1022  75 10011 01024    CLEAR
1023  11 20000 01164      MATRIX
1024  16 01140 00000    SET RERUN
1025  16 01034 01176    9 ⟶DIGIT
1026  76 00000 01016    READ
1027  76 10000 10000       ONE
1030  76 10000 01177           ROW
1031  37 01031 01032    LAST ROW SWITCH
1032  11 01136 01042    PRESET MATRIX STORE
1033  11 01156 01175    SET INDEX = 3
1034  31 01141 00011    SENTINEL ⟶(AL)
1035  32 20000 00003    SL4 (AR) , SL3 (AL)
1036  44 01037 01040    BIT IN THIS COLUMN ?
1037  32 01176 00000    YES : ADD-DIGIT
```

PX 71900-9-(131)

CARD READ ROUTINE

1040	46	01041	01035	REACHED SENTINEL ?
1041	31	20000	00000	REMOVE SENTINEL
1042	00	00000	00000	ADD TO MATRIX WORD
1043	21	01042	01161	STEP MATRIX STORE ORDER
1044	41	01175	01034	ROW WORD EXHAUSTED ?
1045	37	01045	01046	SWITCH
1046	55	01177	00000	ROW WORD 2 → (Q)
1047	37	01045	01033	BCD INFO. → MATRIX
1050	55	01016	00034	ROW WORD 3 → (Q)
1051	37	01045	01034	BCD INFO → MATRIX
1052	37	01052	01053	SWITCH FOR SIGN ROW
1053	41	01176	01026	DIGIT-1→DIGIT, REPEAT
1054	37	01052	01025	STORE 9'S FOR-SIGNS
1055	37	01031	01026	READ LAST ROW
1056	37	01012	01014	SET CONVERSION SWITCH
1057	15	01136	01126	PRESTORE MATRIX TRANSFER
1060	37	01121	01061	SET CONVERSION REPEAT
1061	55	30000	00000	PARAMETER WORD 1 → (Q)
1062	44	01063	01064	LAST FIELD ?
1063	16	01012	01121	SET CONVERSION EXIT
1064	55	10000	00013	SL_{11} (Q)
1065	51	01157	20000	SET
1066	16	20000	01107	SHIFT
1067	55	10000	00006	STORE
1070	51	01157	01175	B
1071	55	10000	00006	STORE
1072	51	01157	01177	L
1073	55	10000	00006	STORE
1074	51	01157	01016	R
1075	32	20000	00016	$R \cdot 2^{15} \rightarrow (A)$
1076	35	01135	01077	SET NEXT INSTRUCTION
1077	00	00000	00000	STORE 10^{R-1}

CARD READ ROUTINE

1100	41	01175 01122	SHIFT B DIGITS
1101	37	01134 01102	CONVERT
1102	41	01177 01123	L TERMS
1103	37	01130 01105	SHIFT DECIMAL POINT
1104	37	01134 01105	CONVERT
1105	41	01016 01123	REMAINING TERMS
1106	54	01077 10107	$1/2 \cdot 10^{R-1} \rightarrow$ ROUNDING TERM
1107	31	01175 30000	$N \cdot 10^{R-1} \cdot 2^{S} \rightarrow (A)$
1110	32	10000 00000	ADD ROUNDING TERM
1111	73	01077 01077	STORE $N \cdot 2^{S}$
1112	37	01130 01123	SHIFT SIGN
1113	51	01163 20000	SIGN DIGIT $\rightarrow$ (A)
1114	47	01115 01116	SIGN NEGATIVE ?
1115	13	01077 01077	YES: $-N \rightarrow N$
1116	11	01077 30000	STORE RESULT
1117	21	01061 01160	ADVANCE B
1120	21	01116 01142	ADVANCE D
1121	45	00000 31121	
1122	16	01137 01130	
1123	41	01176 01127	MATRIX WORD EXHAUSTED ?
1124	11	01163 01176	YES: RESET INDEX
1125	21	01126 01160	STEP TRANSFER
1126	11	30000 01164	TRANSFER NEW MATRIX WORD
1127	55	01164 00004	SHIFT TO NEXT DEC. DIGIT
1130	37	01130 01131	SWITCH
1131	31	01175 00002	$4N \rightarrow (A)$
1132	32	01175 00001	$10N \rightarrow (A)$
1133	52	01162 01175	10N PLUS DIGIT $\rightarrow$ N
1134	45	00000 31134	CONVERSION EXIT
1135	11	01142 01077	PRESET
1136	35	01164 01164	PRESET
1137	00	00000 01100	

PX 71900-9-(131)

CARD READ ROUTINE

```
1140  00 08000 01000
1141  54 00000 00000        MODIFY COMMAND
1142  00 00000 00001        1
1143  00 00000 00001        TABLE
1144  00 00000 00012
1145  00 00000 00144           POWERS
1146  00 00000 01750
1147  00 00000 23420              OF
1150  00 00003 03240                 TEN
1151  00 00036 41100
1152  00 00461 13200
1153  00 05753 60400
1154  00 73465 45000
1155  11 24027 62000
1156  00 00000 00003        3
1157  00 00000 00077        EXTRACTOR
1160  00 00001 00000        U ADVANCE
1161  00 00001 00001        U AND V ADVANCE
1162  00 00000 00017        4 BIT EXTRACTOR
1163  00 00000 00010        8
1164  00 00000 00000        MATRIX WORD      1
1165  00 00000 00000                         2
1166  00 00000 00000                         3
1167  00 00000 00000                         4
1170  00 00000 00000                         5
1171  00 00000 00000                         6
1172  00 00000 00000                         7
1173  00 00000 00000                         8
1174  00 00000 00000                         9
1175  00 00000 00000        INDEX
1176  00 00000 00000        DIGIT
1177  00 00000 00000        ROW WORD 2
```

PX 71900-9-(131)

SOLUTION OF SIMULTANEOUS LINEAR EQUATIONS BY THE METHOD OF CROUT

PART I - INTRODUCTION

This program is applicable to systems of n equations in n unknowns of the form

$$1)\quad \begin{aligned} a_{11}X_1 + a_{12}X_2 + \cdot \quad \cdot \quad \cdot + a_{1n}X_n &= a_{1\ n+1} \\ a_{21}X_2 + a_{22}X_2 + \cdot \quad \cdot \quad \cdot + a_{2n}X_n &= a_{2\ n+1} \\ \cdot \quad \cdot \quad \cdot \quad \cdot \qquad\qquad \cdot \quad \cdot \quad & \cdot \\ a_{n1}X_1 + a_{n2}X_2 + \cdot \quad \cdot \quad \cdot + a_{nn}X_n &= a_{n\ n+1} \end{aligned}$$

where $n \leq 31$.

This routine employs the floating point two-register number representation and the corresponding Arithmetic Package CA001.

PART II - ANALYSIS

In addition to the given (augmented) matrix $[M]$

$$2)\quad \begin{matrix} a_{11} & a_{12} & \cdot & \cdot & \cdot & a_{1\ n+1} \\ a_{21} & a_{22} & \cdot & \cdot & \cdot & a_{2\ n+1} \\ \cdot & \cdot & & & & \cdot \\ a_{n1} & a_{n2} & \cdot & \cdot & \cdot & a_{n\ n+1} \end{matrix}$$

the method requires the formation of an auxiliary matrix $[\bar{M}]$

$$3)\quad \begin{matrix} \bar{a}_{11} & \bar{a}_{12} & \cdot & \cdot & \cdot & \bar{a}_{1\ n+1} \\ \bar{a}_{21} & \bar{a}_{22} & \cdot & \cdot & \cdot & \bar{a}_{2\ n+1} \\ \cdot & \cdot & & & & \cdot \\ \bar{a}_{n1} & \bar{a}_{n2} & \cdot & \cdot & \cdot & \bar{a}_{n\ n+1} \end{matrix}$$

which is used in generating the solutions

$$X_1,\ X_2 \quad \cdot \quad \cdot \quad \cdot \quad X_n$$

The elements of the auxiliary matrix are determined by

$$4)\quad \begin{aligned} \bar{a}_{i1} &= a_{i1} \qquad i = 1, 2, \cdot \cdot \cdot\ n \\ \bar{a}_{1j} &= a_{1j}/\bar{a}_{11} \qquad j = 2, 3, \cdot \cdot \cdot\ n+1 \\ \bar{a}_{ij} &= a_{ij} - \sum_{k=1}^{j-1} \bar{a}_{ik}\bar{a}_{kj}, \qquad 2 \leq j \leq i \leq n \\ \bar{a}_{ij} &= \Big(a_{ij} - \sum_{k=1}^{i-1} \bar{a}_{ik}\bar{a}_{kj}\Big)/\bar{a}_{ii}, \qquad 2 \leq i < j \leq n+1 \end{aligned}$$

The solutions are determined by

$$5)\quad X_i = \bar{a}_{i\,n+1} - \sum_{k=i+1}^{n} a_{ik}X_k, \qquad i = 1, 2, \cdot \cdot \cdot, n$$

The steps which lead to 4) and 5) are described in detail in section on method of elimination in the chapter on simultaneous linear equations in <u>Numerical Calculus</u>, by Milne.

The solutions are improved by a recursive procedure. Suppose the solutions first obtained by use of 5) are denoted by ${}^0X_i$. Then modified system

PX 71900-9-(132)

$$a_{11}X_1 + a_{12}X_2 + \cdots + a_{1n}X_n = a_{1\,n+1} - \sum_{k=1}^{n} a_{1k}\,{}^{0}X_k$$
$$a_{21}X_1 + a_{22}X_2 + \cdots + a_{2n}X_n = a_{2\,n+1} - \sum_{k=1}^{n} a_{2k}\,{}^{0}X_k$$
$$\cdots \qquad \cdots \qquad \cdots$$
$$a_{n1}X_1 + a_{n2}X_2 + \cdots + a_{nn}X_n = a_{n\,n+1} - \sum_{k=1}^{n} a_{nk}\,{}^{0}X_k$$

is solved to obtain a first set of corrections ${}^{1}X_i$.
Then modified system

$$a_{11}X_1 + a_{12}X_2 + \cdots + a_{1n}X_n = a_{1\,n+1} - \sum_{k=1}^{n} a_{1k}\,({}^{0}X_k + {}^{1}X_k)$$
$$a_{21}X_1 + a_{22}X_2 + \cdots + a_{2n}X_n = a_{2\,n+1} - \sum_{k=1}^{n} a_{2k}\,({}^{0}X_k + {}^{1}X_k)$$
$$\cdots \qquad \cdots \qquad \cdots$$
$$a_{n1}X_1 + a_{n2}X_2 + \cdots + a_{nn}X_n = a_{n\,n+1} - \sum_{k=1}^{n} a_{nk}\,({}^{0}X_k + {}^{1}X_k)$$

is solved to obtain a second set of corrections ${}^{0}X_i$. When the right members (differences) of the modified systems become tolerably close to zero, the final solutions are obtained by adding the corrections to ${}^{0}X_i$. This program provides for five determinations of differences by setting an appropriate difference iteration index register.

PART III - PROGRAMMING

PROGRAM STEPS

1. PT loads the routines, except card read, onto the drum. Card read goes to ES.
2. An image of the card read routine is stored on the drum.
3. The cards are read onto the drum.
4. An exit from the card read routine is executed at (PAK) = 337.
5. The main routine and arithmetic package are read into ES.
6. The computation of 4) begins.
7. Problem ends and is set up to begin the solution of another system of equations.

INPUT

Header card: It is to contain only n (the number of rows of the given matrix) punched in columns 5 and 6 with the remaining columns blank. N (Required by IC003) is computed.

Data cards: Given matrix is punched by rows with a new card to begin each row. (Extra zeros, which are used to fill the cards in keeping with the above requirement, are deleted after read-in.) Refer to IC003-1.

OUTPUT

The final solutions and their associated differences are punched in cards each time computed. Refer to IC004.

PART IV - OPERATIONAL PROCEDURE

1. Place input cards in read hopper, blank cards in punch hopper.
2. PT load program.
3. MD start (40000).
4. Final stop.

MANUAL STOPS

MS 1 For debugging purposes
MS 2 For debugging purposes
MS 3 To stop problem after each row of the auxiliary matrix is computed.

FAILURES

In case of a failure described below, start the problem again.

Causes;

1. An attempted division by zero may occur with (PAC)=363 if any diagonal element of the auxiliary matrix is zero. Interchange a pair of rows of the given matrix (interchange the appropriate input cards) and try again.

2. MCT or SCC fault.

PROGRAM SAMPLER

Automatic blotto dumps of data regions are provided.

START	DUMPS
00630	Row of given matrix in ES at sample time.
00633	Column of auxiliary matrix in ES at sample time.
00636	[M] storage by rows.
00645	[M] storage by columns.

STORAGE ASSIGNMENTS

Given matrix	40001-52270
(A row of the given matrix)	01600-01777
Auxiliary matrix	56100-70437
(A column of the auxiliary matrix)	01400-01577
Final solutions	76500-76677
Main routine	77250-77775
Difference iteration index register	77764
Arithmetic Package	77125-77251
ES image (stored during punching)	74001-75777
Corrections to answers	01200-01377
Punch routine	70440-71207
Storage of original ES data	71220-73226
Available	52271-56077

CROUT CN 005

```
40000          45 00000 71220    START'
71220          75 31777 71222    STORE CARD INPUT RT.
71221          11 00001 71230
71222          75 31777 71224    TRANSFER CARD INPUT RT.
71223          11 71230 00001      TO ES
71224          11 71226 40000    SET FOR NEXT MATRIX READ
71225          45 00000 00300    JUMP TO READ MATRIX
71226          45 00000 71222    PRESET
                                       STORAGE
77250  00100   00 00000 00000    i
77251  00101   00 00000 00000    n
77252  00102   00 00000 00000    (30000+2n+2) · 2^15
77253  00103   00 00000 00000    (30000+2n) · 2^15
77254  00104   00 00000 00000    j
77255  00105   00 00000 00000    [2n(i-1)+2i-1] · 2^15
77256  00106   00 00000 00000    BLANK
77257  00107   00 00000 00000    k'
77260  00110   00 00000 00000    k
77261  00111   00 00000 00000    Σ1
77262  00112   00 00000 00000    Σ2
77263  00113   00 00000 00000    n INDEX
77264  00114   00 00000 00000    2n
77265  00115   00 00000 00000    2n · 2^15
77266  00116   00 00000 00000    (2j-1) · 2^15
77267  00117   00 00000 00000    (2n+1) · 2^15
77270  00120   00 00000 00000    {n/5}
77271  00121   00 00000 00000    5n({n/5}+1) = N
77272  00122   00 00000 00000    2N/n · 2^15
77273  00123   00 00000 00000    2n+2
77274  00124   00 00000 00000    BLANK
77275  00125   00 00000 00000    (n-2) INDEX
77276  00126   00 00000 00000    (n-1)
```

CROUT

```
77277  00127  00 00000 00000
77300  00130  00 00000 00000
77301  00131  00 00000 00000
77302  00132  00 00000 00000
77303  00133  00 00000 00000
77304  00134  00 00000 00000
77305  00135  00 00000 00000
77306  00136  00 00000 00000
77307  00137  11 01577 00025          PRESETS
77310  00140  00 00002 00000
77311  00141  00 30000 00000
77312  00142  11 40000 01600
77313  00143  11 56077 01400
77314  00144  11 01577 00111
77315  00145  11 01577 00025
77316  00146  11 01377 00027
77317  00147  00 00000 00344
77320  00150  11 00111 01577
77321  00151  11 01577 00027
77322  00152  11 01600 56077
77323  00153  11 56077 01600
77324  00154  11 00111 01377
77325  00155  00 00000 01200
77326  00156  11 01377 00025
77327  00157  11 01577 00027
77330  00160  75 30530 77332     ROUTINE ——→FS
77331  00161  11 77250 00100
77332  00162  75 30050 00164     EMERGENCY PUNCH-OUT ——→FS
77333  00163  11 77040 00630
77334  00164  75 30105 00166     ARITHMETIC PACKAGE ——→FS
77335  00165  11 77125 01000
77336  00166  11 00140 20000     SET
```

PX 71900-9-(132)

CROUT

```
77337  00167  72 00140 00101
77340  00170  35 00141 20000          ROW
77341  00171  15 20000 00214
77342  00172  15 20000 00216              TRANSFERS
77343  00173  71 00101 00224
77344  00174  11 20000 00010
77345  00175  11 00101 20000
77346  00176  73 00224 20000
77347  00177  35 00074 20000      COMPUTE N
77350  00200  71 20000 00010
77351  00201  11 20000 00121
77352  00202  31 00121 00020      NUMBER OF CELLS PER ROW
77353  00203  73 00101 00122      INCLUDING EXTRA ZEROS
77354  00204  71 00041 00101
77355  00205  35 00041 00123      NUMBER OF CELLS PER ROW
77356  00206  11 00101 20000
77357  00207  36 00041 00125      INDEX FOR MATRIX [M] TRANSFER
77360  00210  11 00122 20000      PRESETS
77361  00211  35 00215 00215        TO DELETE
77362  00212  11 00123 20000          EXTRA
77363  00213  35 00217 00217            ZEROS
77364  00214  75 00000 00216      TRANSFER ROW OF [M] TO ES
77365  00215  11 40001 01600
77366  00216  75 00000 00220      TRANSFER ROW OF [M] TO DRUM
77367  00217  11 01600 40001        EXTRA ZEROS DELETED
77370  00220  21 00215 00122      STEP U ADDRESS
77371  00221  21 00217 00123      STEP V ADDRESS
77372  00222  41 00125 00214      ALL ROWS TRANSFERRED? NO: RETURN
77373  00223  56 10000 00227      YES: TAKE 227
77374  00224  00 00000 00005      5
77375  00225  00 00000 00000      BLANK
77376  00226  00 00000 00000      BLANK
```

CROUT

```
77377  00227  75 10200 00231     CLEAR ANSWER STORAGE
77400  00230  11 00040 76500
77401  00231  16 00623 00533     SET SWITCH III
77402  00232  71 00101 00041
77403  00233  11 20000 00114     CLEAR
77404  00234  31 00114 00017
77405  00235  11 20000 00115       [M̄]
77406  00236  11 00115 20000
77407  00237  35 00073 00117         MATRIX
77410  00240  75 17777 00242
77411  00241  11 00040 56100           STORAGE
77412  00242  11 00074 00100     SET ROW INDEX
77413  00243  11 00117 20000     SET
77414  00244  35 00073 20000       TO
77415  00245  35 00141 00102         TRANSFER
77416  00246  11 00115 20000           ROW
77417  00247  35 00141 00103             AND
77420  00250  15 00102 00265               COLUMN
77421  00251  15 00103 00276                 TO
77422  00252  15 00103 00422                   ES
77423  00253  15 00102 00462
77424  00254  16 00147 00316     SET SWITCH Ia
77425  00255  11 00074 00104     SET COLUMN INDEX
77426  00256  71 00100 00140
77427  00257  36 00140 20000     PRESET
77430  00260  71 20000 00101       DRUM
77431  00261  36 00073 20000         ADDRESS
77432  00262  72 00140 00100           OF
77433  00263  11 20000 00105             ROW
77434  00264  35 00142 00266
77435  00265  75 00000 00267     TRANSFER ROW
77436  00266  00 00000 00000       OF [M] TO ES
```

PX 71900-9-(132)

CROUT

```
77437  00267  56 10000 00270
77440  00270  11 00104 20000
77441  00271  36 00074 20000     PRESET
77442  00272  71 20000 00101       DRUM
77443  00273  71 20000 00140         ADDRESS
77444  00274  35 00073 20000           OF
77445  00275  35 00143 00277             COLUMN
77446  00276  75 00000 00300     TRANSFER COLUMN
77447  00277  00 00000 00000     OF [M̄] TO ES
77450  00300  56 20000 00301
77451  00301  11 00100 20000     ELEMENT TO RIGHT OF DIAGONAL ?
77452  00302  42 00104 00306     YES: TAKE 306
77453  00303  11 00104 20000     NO: SET K'
77454  00304  36 00074 00107
77455  00305  45 00000 00307
77456  00306  36 00074 00107     YES: SET K'
77457  00307  11 00074 00110     SET K = 1
77460  00310  71 00104 00140     TRANSFER
77461  00311  36 00073 20000
77462  00312  35 00144 00314       ELEMENT OF [M]
77463  00313  75 30002 00315
77464  00314  00 00000 00000         INTO (111, 112)
77465  00315  11 00107 20000     K : K'  >  TAKE 344
77466  00316  42 00110 30000            ≤  TAKE 317
77467  00317  71 00110 00140     PRESET ADDRESS OF
77470  00320  36 00073 20000       ā_iK
77471  00321  35 00145 00323
77472  00322  75 30002 00324
77473  00323  00 00000 00000     ā_iK ⟶ (25, 26)
77474  00324  71 00110 00140     TRANSFER
77475  00325  36 00073 20000
77476  00326  35 00146 00330       ā_Kj
```

CROUT

77477	00327	75	30002	00331	
77500	00330	00	00000	00000	INTO (27, 30)
77501	00331	37	01001	01003	PRODUCT OF ELEMENTS
77502	00332	13	00031	00027	
77503	00333	11	00032	00030	
77504	00334	11	00111	00025	$a_{ij} \longrightarrow (25, 26)$
77505	00335	11	00112	00026	
77506	00336	37	01001	01002	$a_{ij} - \bar{a}_{iK}\ \bar{a}_{Ki} = \Sigma$
77507	00337	11	00031	00111	$\Sigma \longrightarrow (111, 112)$
77510	00340	11	00032	00112	
77511	00341	21	00110	00074	STEP K
77512	00342	45	00000	00315	RETURN
77513	00343	00	00000	00000	BLANK
77514	00344	11	00100	20000	$j : i >$ TAKE 354
77515	00345	42	00104	00354	$\leqslant$ STORE RESULT
77516	00346	71	00104	00041	PRESET ES
77517	00347	36	00074	20000	ADDRESS FOR
77520	00350	35	00150	00352	$[\bar{M}]$ STORAGE
77521	00351	75	30002	00353	
77522	00352	00	00000	00000	STORE RESULT
77523	00353	45	00000	00367	
77524	00354	71	00100	00140	DIVIDE
77525	00355	36	00073	20000	
77526	00356	35	00151	00360	BY
77527	00357	75	30002	00361	
77530	00360	00	00000	00000	DIAGONAL
77531	00361	11	00111	00025	
77532	00362	11	00112	00026	ELEMENT
77533	00363	37	01001	01004	OF $[\bar{M}]$
77534	00364	11	00031	00111	STORE
77535	00365	11	00032	00112	RESULT
77536	00366	45	00000	00346	RETURN

PX 71900-9-(132)

CROUT

```
77537  00367  11 00101 20000    ROW
77540  00370  35 00074 20000      USED
77541  00371  43 00104 00374        UP?  YES: TAKE 374
77542  00372  21 00104 00074    NO:  SET NEXT COL.
77543  00373  45 00000 00267    RETURN
77544  00374  11 00101 00113    STORE
77545  00375  71 00100 00041      ELEMENTS
77546  00376  36 00074 20000        OF [M̄]
77547  00377  35 00152 00401          ON
77550  00400  75 30002 00402           DRUM
77551  00401  00 00000 00000
77552  00402  21 00401 00114    STEP STORAGE
77553  00403  21 00401 00140      ADDRESSES
77554  00404  41 00113 00400    ROW STORED?  NO: TAKE 400
77555  00405  11 00101 20000    YES:  n ⟶ (A)
77556  00406  43 00100 00411    LAST ROW? YES: TAKE 411
77557  00407  21 00100 00074    NO: STEP ROW SELECTION
77560  00410  56 30000 00255    RETURN TO COMPUTE NEXT ROW
77561  00411  56 30000 00412    YES: COMPUTE FINAL MATRIX
77562  00412  11 00101 00100    SET UP
77563  00413  11 00101 20000      TO COMPUTE
77564  00414  35 00074 00104        FINAL MATRIX
77565  00415  11 00101 00107
77566  00416  71 00101 00101    TRANSFER
77567  00417  54 20000 00020
77570  00420  35 00073 20000      n + 1 st
77571  00421  35 00143 00423
77572  00422  75 00000 00424        COL.
77573  00423  00 00000 00000          INTO ES
77574  00424  11 00101 00113    SET TFR INDEX
77575  00425  71 00100 00140    TRANSFER
77576  00426  36 00073 20000      ITH COL.
```

PX 71900-9-(E32)

CROUT

```
77577  00427  35 00153 00431        INTO ES
77600  00430  75 30002 00432          BY ROWS
77601  00431  00 00000 00000
77602  00432  21 00431 00115    STEP U ADDRESS
77603  00433  21 00431 00041    STEP V ADDRESS
77604  00434  41 00113 00430    ROW COMPLETE? NO: TAKE 430
77605  00435  11 00100 20000    YES:
77606  00436  35 00074 00110
77607  00437  37 00316 00310    SET I_b
77610  00440  71 00100 00041    STORE
77611  00441  36 00074 20000
77612  00442  35 00154 00444      X_i
77613  00443  75 30002 00445
77614  00444  00 00000 00000        AT ES ADDRESS
77615  00445  23 00100 00074
77616  00446  47 00424 00447    LAST X_i COMPUTED? NO: TAKE 424
77617  00447  45 00000 00534    YES: TAKE 534
77620  00450  37 70440 70441    PUNCH ANSWERS
77621  00451  00 76500 30000
77622  00452  11 00074 00100    TRANSFER
77623  00453  11 00074 00104
77624  00454  71 00100 00140      ROW
77625  00455  36 00140 20000
77626  00456  71 20000 00101        OF
77627  00457  72 00140 00100
77630  00460  36 00073 20000            [M]
77631  00461  35 00142 00463
77632  00462  75 00000 00464                INTO
77633  00463  00 00000 00000                  ES
77634  00464  11 00040 00111    CLEAR
77635  00465  11 00040 00112      STORAGE
77636  00466  71 00140 00104    TRANSFER
```

CROUT

```
77637  00467  36 00073 00116      X¿
77640  00470  35 00624 00472        INTO
77641  00471  75 30002 00473          CELLS
77642  00472  00 00000 00000            (25), (26)
77643  00473  11 00116 20000      TRANSFER
77644  00474  35 00157 00476        ELEMENT OF
77645  00475  75 30002 00477          [M] INTO
77646  00476  00 00000 00000            (27), (30)
77647  00477  37 01001 01003      MULTIPLY
77650  00500  11 00111 00025      COMPUTE
77651  00501  11 00112 00026        NEW
77652  00502  11 00031 00027          VALUE
77653  00503  11 00032 00030            FOR
77654  00504  37 01001 01002              THIS ROW
77655  00505  11 00031 00111      STORE
77656  00506  11 00032 00112        Σ
77657  00507  11 00101 20000
77660  00510  43 00104 00513      ROW FINISHED? YES:TAKE 513
77661  00511  21 00104 00074      NO: SET FOR NEXT ELEMENT
77662  00512  45 00000 00466      JUMP TO CONTINUE ROW
77663  00513  11 00117 20000      YES:
77664  00514  35 00137 00516        TRANSFER
77665  00515  75 30002 00517          RH SIDE
77666  00516  00 00000 00000            INTO (25),(26)
77667  00517  13 00111 00027      COMPUTE
77670  00520  11 00112 00030        THE
77671  00521  37 01001 01002          DIFFERENCES
77672  00522  75 30002 00524      STORE
77673  00523  11 00031 30000        DIFFERENCES AT  1200
77674  00524  11 00101 20000      ALL
77675  00525  43 00100 00531        ROWS FINISHED? YES:TAKE 531
77676  00526  21 00100 00074      NO: STEP
```

PX 71900-9-(132)

CROUT

```
77677  00527  21 00523 00041      STORAGE ADDRESSES
77700  00530  45 00000 00453  JUMP TO TRANSFER ROW
77701  00531  37 70440 70441  YES: PUNCH
77702  00532  00 01200 30000      DIFFERENCES
77703  00533  45 00000 00000  JUMP TO III OR TO FINAL STOP
77704  00534  16 00155 00523  PRESET TO STORE CORRECTIONS
77705  00535  16 00101 00451  PRESET TO PUNCH ANSWERS
77706  00536  16 00101 00532  PRESET TO PUNCH DIFFERENCES
77707  00537  45 00000 00540  DUMMY
77710  00540  11 00101 20000  SET
77711  00541  36 00074 00126      INDEX
77712  00542  15 00617 00546  PRESET ADDRESS OF ANSWERS
77713  00543  15 00621 00550  PRESET ADDRESS OF CORRECTIONS
77714  00544  16 00622 00553  PRESET TO STORE ANSWERS
77715  00545  75 30002 00547  COMPUTE
77716  00546  11 76500 00025
77717  00547  75 30002 00551      CORRECTED
77720  00550  11 01400 00027
77721  00551  37 01001 01002          ANSWER
77722  00552  75 30002 00554
77723  00553  11 00031 76500  STORE ANSWER
77724  00554  21 00546 00140  STEP ADDRESSES
77725  00555  21 00550 00140      TO COMPUTE
77726  00556  21 00553 00041          NEXT ANSWER
77727  00557  41 00126 00545  ALL ANSWERS COMPUTED? NO: TAKE 545
77730  00560  45 00000 00450  YES: JUMP TO PUNCH ANSWERS
77731  00561  57 00000 00000  FINAL STOP
77732  00562  71 00101 00101  TRANSFER
77733  00563  35 20000 20000
77734  00564  35 00074 20000      DIFFERENCES
77735  00565  35 00620 00567
77736  00566  75 30200 00570          TO DRUM
```

PX 71900-9-(132)

CROUT

```
77737  00567  11 01200 56077
77740  00570  41 00614 00572    INDEX FOR FINAL STOP
77741  00571  16 00613 00533    PRESET FINAL STOP
77742  00572  75 30200 00574    TRANSFER COLUMN OF
77743  00573  11 01200 01400      DIFFERENCES INTO (1400)
77744  00574  11 00074 00100    SET τ =1
77745  00575  16 00147 00316    PRESET Ia
77746  00576  16 00617 00410    PRESET IIb
77747  00577  11 00101 20000    SET
77750  00600  35 00074 00104      TRANSFER
77751  00601  11 00101 00113        INDEX
77752  00602  71 00100 00140    TRANSFER
77753  00603  36 00073 20000
77754  00604  35 00616 00606      TRANSPOSE OF
77755  00605  75 30002 00607
77756  00606  11 56077 01600        DIFFERENCES
77757  00607  21 00606 00041
77760  00610  21 00606 00115          INTO (1600 . . .)
77761  00611  41 00113 00605    FINISHED TRANSFER? NO: TAKE 605
77762  00612  45 00000 00267    YES: BEGIN NEXT ITERATION
77763  00613  45 00000 00561    PRESET  FINAL STOP
77764  00614  00 00000 00003    ITERATIONS INDEX
77765  00615  00 00000 00000    BLANK
77766  00616  11 56077 01600    PRESETS
77767  00617  00 76500 00601
77770  00620  11 01200 56077
77771  00621  00 01400 00613
77772  00622  00 00000 76500
77773  00623  00 00000 00562
77774  00624  11 76477 00025
77775  00625  75 00002 00016
77776  00626  17 00000 77565
```

CROUT

```
77777  00627  45 00000 00175
                    EMERGENCY PUNCH-OUT
77040  00630  37 70440 70441
77041  00631  00 01600 00244      ROW STORAGE
77042  00632  56 00000 00633
77043  00633  37 70440 70441
77044  00634  00 01400 00244      COLUMN STORAGE
77045  00635  56 00000 00636
77046  00636  71 00101 00101
77047  00637  35 00101 20000
77050  00640  45 00000 00641      [M ]
77051  00641  16 20000 00643
77052  00642  37 70440 70441
77053  00643  00 40001 00000
77054  00644  56 00000 00645
77055  00645  71 00101 00101
77056  00646  35 00101 20000
77057  00647  45 00000 00650      [M̄ ]
77060  00650  16 20000 00652
77061  00651  37 70440 70441
77062  00652  00 56100 00000
77063  00653  56 00000 00000

77067         11 00102 20000      SET
77070         73 77374 20000
77071         35 00074 20000        UP TO
77072         71 20000 00102
77073         71 20000 77374            READ
77074         11 00102 77251
77075         11 20000 00102              CARDS
77076         11 20000 00107
77077         45 00000 00322
```

PX 71900-9-(132)

SQUARE ROOT-FLOATING POINT
(Single Precision)

Given a single-precision floating point number, $x = N.2^P$ (see CA 001); compute $\sqrt{x} = \sqrt{N'} \cdot 2^{P'/2} = N''.2^{P''}$.

Initial State: (00031) = N

(00032) = P

Final State: (00031) = N''

(00032) = P''

N' and P' are defined as follows:

If P is even; N' = N, P' = P

If P is odd; N' = N/2, P' = P + 1

Formula used: Newtons' Iteration

$$Y_i = x_{i-1} + 1/2 \ (N'/X_{i-1} - X_{i-1})$$

First approximation = $2^{35}-1$ (ṡ)

Convergence is assumed when $\Delta x \gtrsim 0$.

Special tests: If N = 0, set N'' = 0

If N' = ṡ, set N'' = ṡ

Alarm: If N is negative.

Drum Address: 77537 - 77567, $(25)_{10} = (31)_8$

Number of commands for assembly modification $(21)_{10} = (25)_8$

No constants or temporaries used.

Standard form.

FLOATING SQUARE ROOT

77537	01000	37	76000	76002	ALARM EXIT
77540	01001	45	00000	31001	EXIT
77541	01002	11	00031	20000	ENTRANCE, N → A
77542	01003	46	01000	01004	N NEG ? Yes → Alarm
77543	01004	47	01005	01001	No ↓ N:0 = → Exit ≠ ↓
77544	01005	12	00032	10000	
77545	01006	55	10000	00043	P EVEN → 1012
77546	01007	44	01010	01012	P ODD ↓
77547	01010	21	00032	01025	P + 1 → 32 (D')
77550	01011	54	00031	00107	N/2 → 31 (N')
77551	01012	55	00032	00043	D'/2 → 32 (D'')
77552	01013	43	01026	01001	N' : 9 = → Exit ≠ ↓
77553	01014	11	01026	01027	9 (35) → X_{i-1}
77554	01015	31	00031	00042	$\frac{1}{2}N'$ (70)
77555	01016	73	01027	01030	$\frac{1}{2}N'/X_{i-1}$ (35) → 1030, Q
77556	01017	54	01027	00107	$\frac{1}{2}X_{i-1}$ (35) → 1027
77557	01020	23	10000	01027	$\frac{1}{2}(N'/X_{i-1} - X_{i-1}) = \Delta x$ (35) → Q
77560	01021	21	01027	01030	$\frac{1}{2}(X_{i-1} + N'/X_{i-1}) = X_i$ (35) → 1027
77561	01022	44	01015	01023	ΔX:0 ≤ ≥ → 1015 ↓
77562	01023	11	20000	00031	X_i = N'' → 31
77563	01024	45	00000	01001	JUMP TO EXIT
77564	01025	00	00000	00001	1 (0)
77565	01026	37	77777	77777	9 (35)
77566	01027	00	00000	00000	X_{i-1} , $1/2X_{i-1}$, X_i
77567	01030	00	00000	00000	$\frac{1}{2}N'/X_{i-1}$

CUBE ROOT-FLOATING POINT
(Single Precision)

Given a single-precision floating point number, $x = N \cdot 2^p$ (see CA 001);

compute $\sqrt[3]{x} = \sqrt[3]{N'} \cdot 2^{p/3} = N'' \cdot 2^{p''}$

Initial State: (00031) = N

(00032) = p

Final State: (00031) = N''

(00032) = p''

N' and p' are defined as follows:

Divide p by 3. If remainder = 0; N' = N, p' = p

If remainder = 1; N' = N/4 , p' = p+2

If remainder = 2; N' = N/2, p' = p+1

Formula used: Newton's Iteration

$$x_i = x_{i-1} + 1/3 \left(N'/x_{i-1}^2 - x_{i-1} \right)$$

First approximation $2^{35} - 1$ $(\dot{9})$, N' > 0

$1 - 2^{35}$ $(-\dot{9})$, N' < 0

Convergence is assumed when $N' \Delta x \geq 0$.

Special tests: If N = 0, set N'' = 0

If N' = 2^{35-1}, set N'' = 2^{35-1}

If N' = $1-2^{35}$, set N'' = $1-2^{35}$

Drum address: 77570 - 77634, $(37)_{10} = (45)_8$

Number of commands for assembly modification: $(31)_{10} = (37)_8$

No constants or temporaries used.

Standard form.

PX 71900-9-(134)

THE RAMO-WOOLDRIDGE CORPORATION
Los Angeles 45, California

Fixed Point Card Output Subroutine

Specifications

Identification Tag:	CPO-O
Type:	Subroutine
Subroutine Designation:	SUB 51641 21516 (uses 254 consecutive cells when assembled)
Storage:	185 instructions, addresses 88800 thru 88899 88900 thru 88984 30 constants in program, addresses 88985 thru 88999 89000 thru 89014 215 words total program storage Temporary storage used but not stored in program. 254 consecutive cells must be provided to assemble this subroutine. The constant pool and temporary storage pool are used by this routine.
Program Entrance:	88802
Program Exit:	88801
Machine Time:	Card punch speed (see text)
Mode of Operation:	Fixed point

Coded by:	R. Summers	January, 1956
Code Checked by:	R. Beach	February, 1956
Machine Checked by:	R. Beach	February, 1956
Approved by:	W. F. Bauer	March 26, 1956

PX 71900-9-(135)

Description

This routine will output up to four fixed point numbers per card punched. These numbers are specified by parameter words contained in a list, the location of which is given in the word following the RJ to this subroutine. This word specifies the first and last address of the parameter list.

For each output word a parameter word is required which has the form:

xx uuuuu vvvvv

where

xx specifies the binary exponent
uuuuu gives the location of the output word
vvvvv gives the address to be punched on the card and associated with the output word.

This information suffices to convert the number to "floating decimal form", i.e. a signed ten digit, normalized rounded fraction with a signed two digit decimal exponent. In addition, a two digit binary exponent and a five digit decimal address are punched. This yields a card output which meets the specifications of the card form for the SNAP RD command, CRI-2 and CPO-1.

There is no restriction on the length of the parameter list. In case there are not enough words to complete a card, blanks will be left in the number fields not used. Though the subroutine must be executed in ES, the control word, parameter list and output words may be in ES or MD. However, to obtain full card punch speed, the parameter words and output words must be in ES. In addition, successive entries to the routine at intervals of 8ms or less will not interrupt the card cycle. Hence, it is possible to employ four word parameter lists which are successively modified by the program between re-entries to this punch routine.

Since 6 bits are allowed to specify the binary exponent XX, the range of scale factors (S) must be such that $0 \leq S \leq 63$.

Programming Instructions

1. Enter subroutine with

RJ 00M01 00M02
yy F0000 L0000

where

00M00 is the location of the first word of this subroutine
F0000 is the location of the first word of the parameter list
L0000 is the location of the last word of the parameter list
yy may have any value.

2. Furnish parameter list.

   This list is composed of parameter words of the form

   xx uuuuu vvvvv

   where

   xx is the binary scale factor associated with the output word

   uuuuu is the location of the output word

   vvvvv is the address or identification associated with the output number and which is placed on the card.

3. Control is returned to the word following the control word after punching.

4. Routine assumes cards are positioned on the punch side of the Bull before entry.

Accuracy

All numbers with binary exponents less than or equal to 35 are represented as ten digit decimal numbers (rounded). All digits of the binary number are used in the conversion so that the result is as accurate as possible. Numbers with binary exponents greater than 35 may have an inaccuracy in the least significant decimal digit since an intermediate binary number (which may be in error in its least significant bit) is used in the conversion. All integers in the range from 0 to 10^{10} are converted exactly.

```
D          88800 51641
D          88900 51741
D          89000 51841
D          08800 01024
D          08900 01124
D          09000 01224
88800   00 00000 00000
88801   MJ 00000 00000
88802   TV 08801 08803
88803   MP 00016 00000
88804   TU A0000 08814
88805   TP A0000 00031
88806   LQ 00031 10021
88807   TV Q0000 00031
88808   ST 00031 00031
88809   TV 08994 08998
88810   TP 09000 09017
88811   TV 08993 08816
88812   TU 08820 08815
88813   RP 30004 08815
88814   TP 00000 00023
88815   TU 00000 08816
88816   TP 00000 00000
88817   IJ 00031 08821
88818   RA 08801 00016
88819   TV A0000 08998
88820   MJ 00023 08824
88821   RA 08815 00015
88822   RA 08816 00016
88823   IJ 09017 08815
88824   EF 00027 08846
88825   RP 20036 08827
88826   CC 09018 A0000
88827   TP 09000 00032
88828   RA 08815 09001
88829   TU 08817 08850
88830   TU 08824 08843
88831   TP 00016 09016
88832   TP 09000 09015
88833   TP 08987 08941
88834   TU 08987 08985
88835   TU 08994 08983
88836   RS 08850 00015
88837   RS 08843 00015
88838   TV 08999 08985
88839   TV 08999 08983
88840   IJ 09017 08843
88841   TV 08831 08985
88842   TV 08831 08983
88843   TP 00000 Q0000
```

```
66571   00 00000 00000
66735   00 00000 00000
67101   00 00000 00000
02000   00 00000 00000
02144   00 00000 00000
02310   00 00000 00000
66571   00 00000 00000
66572   45 00000 00000
66573   16 02001 02003
66574   71 00020 00000
66575   15 20000 02016
66576   11 20000 00037
66577   55 00037 10025
66600   16 10000 00037
66601   36 00037 00037
66602   16 02302 02306
66603   11 02310 02331
66604   16 02301 02020
66605   15 02024 02017
66606   75 30004 02017
66607   11 00000 00027
66610   15 00000 02020
66611   11 00000 00000
66612   41 00037 02025
66613   21 02001 00020
66614   16 20000 02306
66615   45 00027 02030
66616   21 02017 00017
66617   21 02020 00020
66620   41 02331 02017
66621   17 00033 02056
66622   75 20044 02033
66623   27 02332 20000
66624   11 02310 00040
66625   21 02017 02311
66626   15 02021 02062
66627   15 02030 02053
66630   11 00020 02330
66631   11 02310 02327
66632   11 02273 02215
66633   15 02273 02271
66634   15 02302 02267
66635   23 02062 00017
66636   23 02053 00017
66637   16 02307 02271
66640   16 02307 02267
66641   41 02331 02053
66642   16 02037 02271
66643   16 02037 02267
66644   11 00000 10000
```

PX 71900-9-(135)

```
88844   QT 09002 00026
88845   SP Q0000 00005
88846   TP B0000 Q0074
88847   TP Q0000 08895
88848   TP 00016 08936
88849   RJ 08982 08975
88850   TP 00000 A0000
88851   TP A0000 00030
88852   ZJ 08855 08853
88853   RP 10010 08923
88854   TU 00013 09043
88855   TP 08995 08903
88856   SP 09004 00000
88857   TJ 08895 08875
88858   SS 08895 00000
88859   TV A0000 08862
88860   TV 08988 08870
88861   TM 00030 A0000
88862   LA A0000 00000
88863   TP A0000 08895
88864   TP 00013 08936
88865   TP B0000 A0000
88866   ZJ 08867 08961
88867   TP 08996 08903
88868   DV 09003 Q0000
88869   SP A0000 00015
88870   TU A0000 00000
88871   RS 08870 00016
88872   RA 08936 00016
88873   SP Q0000 00000
88874   ZJ 08868 08961
88875   TP 00013 08979
88876   SF 00030 08979
88877   TN 08979 A0000
88878   ZJ 08880 08879
88879   TN 00018 A0000
88880   SA 09005 00000
88881   SA 08895 00000
88882   MP A0000 09006
88883   TP B0000 A0000
88884   TJ 09003 08886
88885   TP 09003 A0000
88886   TN A0000 08936
88887   SP A0000 00015
88888   TU A0000 08892
88889   TN 08895 A0000
88890   AT 09007 08895
88891   SP 00016 00000
88892   RP 00000 08894
88893   MP A0000 09003
```

```
66645   51 02312 00032
66646   31 10000 00005
66647   11 30000 10112
66650   11 10000 02137
66651   11 00020 02210
66652   37 02266 02257
66653   11 00000 20000
66654   11 20000 00036
66655   47 02067 02065
66656   75 10012 02173
66657   15 00015 02363
66660   11 02303 02147
66661   31 02314 00000
66662   42 02137 02113
66663   34 02137 00000
66664   16 20000 02076
66665   16 02274 02106
66666   12 00036 20000
66667   54 20000 00000
66670   11 20000 02137
66671   11 00015 02210
66672   11 30000 20000
66673   47 02103 02241
66674   11 02304 02147
66675   73 02313 10000
66676   31 20000 00017
66677   15 20000 00000
66700   23 02106 00020
66701   21 02210 00020
66702   31 10000 00000
66703   47 02104 02241
66704   11 00015 02263
66705   74 00036 02263
66706   13 02263 20000
66707   47 02120 02117
66710   13 00022 20000
66711   32 02315 00000
66712   32 02137 00000
66713   71 20000 02316
66714   11 30000 20000
66715   42 02313 02126
66716   11 02313 20000
66717   13 20000 02210
66720   31 20000 00017
66721   15 20000 02134
66722   13 02137 20000
66723   35 02317 02137
66724   31 00020 00000
66725   75 00000 02136
66726   71 20000 02313
```

PX 71900-9-(135)

```
88894   MP A0000 00030
88895   00 00000 00000
88896   TP 08986 08907
88897   TM A0000 Q0000
88898   QT 09008 Q0000
88899   SP Q0000 00002
88900   SA Q0000 00001
88901   TP A0000 Q0000
88902   SP B0000 00015
88903   00 00000 00000
88904   RS 08936 00016
88905   MJ 00000 08898
88906   TP 08996 08903
88907   00 00000 00000
88908   RA 08907 00016
88909   TJ 08988 08898
88910   QT 09009 A0000
88911   ZJ 08912 08922
88912   TP 00021 Q0000
88913   TU 08969 08916
88914   RS 08916 00015
88915   TJ 08989 08920
88916   RA 00000 00015
88917   QT A0000 A0000
88918   EJ 09014 08914
88919   MJ 00000 08922
88920   TU 00015 09043
88921   RA 08936 00016
88922   TP 08936 A0000
88923   TM A0000 Q0000
88924   SJ 08925 08926
88925   RJ 08984 08983
88926   TP 00016 08936
88927   RJ 08982 08975
88928   TP 00030 A0000
88929   SJ 08930 08931
88930   RJ 08984 08983
88931   TP 00021 Q0000
88932   TU 08993 08934
88933   TP 08972 08979
88934   QT 00000 A0000
88935   AT 08985 08936
88936   00 00000 00000
88937   RS 08934 00015
88938   LA 09016 00001
88939   IJ 09015 08948
88940   TU 08956 09053
88941   00 00000 00000
88942   RS 08941 09011
```

```
66727   71 20000 00036
66730   00 00000 00000
66731   11 02272 02153
66732   12 20000 10000
66733   51 02320 10000
66734   31 10000 00002
66735   32 10000 00001
66736   11 20000 10000
66737   31 30000 00017
66740   00 00000 00000
66741   23 02210 00020
66742   45 00000 02142
66743   11 02304 02147
66744   00 00000 00000
66745   21 02153 00020
66746   42 02274 02142
66747   51 02321 20000
66750   47 02160 02172
66751   11 00025 10000
66752   15 02251 02164
66753   23 02164 00017
66754   42 02275 02170
66755   21 00000 00017
66756   51 20000 20000
66757   43 02326 02162
66760   45 00000 02172
66761   15 00017 02363
66762   21 02210 00020
66763   11 02210 20000
66764   12 20000 10000
66765   46 02175 02176
66766   37 02270 02267
66767   11 00020 02210
66770   37 02266 02257
66771   11 00036 20000
66772   46 02202 02203
66773   37 02270 02267
66774   11 00025 10000
66775   15 02301 02206
66776   11 02254 02263
66777   51 00000 20000
67000   35 02271 02210
67001   00 00000 00000
67002   23 02206 00017
67003   54 02330 00001
67004   41 02327 02224
67005   15 02234 02375
67006   00 00000 00000
67007   23 02215 02323
```

PX 71900-9-(135)

88943	TJ	08990	08949
88944	RS	08985	09010
88945	RS	08983	09010
88946	TP	00016	09016
88947	TP	09012	09015
88948	IJ	08979	08934
88949	TP	09013	08936
88950	SP	00026	00000
88951	RJ	08982	08976
88952	IJ	00032	08836
88953	RP	30003	08955
88954	TV	08990	08956
88955	TP	09003	08979
88956	EW	00000	00000
88957	EW	10000	00000
88958	EW	10000	00000
88959	RP	20003	08997
88960	RS	08956	00016
88961	SP	08870	00015
88962	TU	A0000	08964
88963	RP	30011	08965
88964	TU	00000	09042
88965	TP	08895	Q0000
88966	SP	08936	00000
88967	EJ	09003	08910
88968	TJ	09002	08973
88969	TP	09053	A0000
88970	TJ	08991	08922
88971	MJ	00000	08912
88972	00	00000	00009
88973	AT	08986	08907
88974	MJ	00000	08898
88975	SP	Q0000	00000
88976	DV	09003	Q0000
88977	SP	A0000	00015
88978	AT	08985	08979
88979	00	00000	00000
88980	LA	09016	00001
88981	IJ	08936	08975
88982	MJ	00000	00000
88983	CC	00000	00000
88984	MJ	00000	00000
88985	CC	00000	00000
88986	TU	A0000	09043
88987	CC	09043	09053
88988	TU	A0000	09053
88989	RA	09044	00015
88990	CC	09012	09052
88991	00	00005	09028

67010	42	02276	02225
67011	23	02271	02322
67012	23	02267	02322
67013	11	00020	02330
67014	11	02324	02327
67015	41	02263	02206
67016	11	02325	02210
67017	31	00032	00000
67020	37	02266	02260
67021	41	00040	02044
67022	75	30003	02233
67023	16	02276	02234
67024	11	02313	02263
67025	77	00000	00000
67026	77	10000	00000
67027	77	10000	00000
67030	75	20003	02305
67031	23	02234	00020
67032	31	02106	00017
67033	15	20000	02244
67034	75	30013	02245
67035	15	00000	02362
67036	11	02137	10000
67037	31	02210	00000
67040	43	02313	02156
67041	42	02313	02255
67042	11	02375	20000
67043	42	02277	02172
67044	45	00000	02160
67045	00	00000	00011
67046	35	02272	02153
67047	45	00000	02142
67050	31	10000	00000
67051	73	02313	10000
67052	31	20000	00017
67053	35	02271	02263
67054	00	00000	00000
67055	54	02330	00001
67056	41	02210	02257
67057	45	00000	00000
67060	27	00000	00000
67061	45	00000	00000
67062	27	00000	00000
67063	11	20000	02363
67064	27	02363	02375
67065	11	20000	02375
67066	21	02364	00017
67067	27	02324	02374
67070	00	00005	02344

PX 71900-9-(135)

```
88992   00 00000 09040
88993   00 09052 00027
88994   00 09042 08811
88995   ZJ 08906 08904
88996   MJ 00000 08907
88997   IJ 08979 08956
88998   RP 20003 00000
88999   EW 44095 00013
89000   00 00000 00003
89001   00 00004 00000
89002   00 00000 77777 B
89003   00 00000 00010
89004   00 00000 00035
89005   00 00000 00037
89006   11 50404 65025 B
89007   54 20000 00153 B
89008   37 77777 77777 B
89009   20 00000 00000 B
89010   00 00012 00000
89011   00 00012 00012
89012   00 00000 00017
89013   00 00000 00004
89014   00 00010 00000
```

```
67071   00 00000 02360
67072   00 02374 00033
67073   00 02362 02013
67074   47 02152 02150
67075   45 00000 02153
67076   41 02263 02234
67077   75 20003 00000
67100   77 47777 00015
67101   00 00000 00003
67102   00 00004 00000
67103   00 00000 77777
67104   00 00000 00012
67105   00 00000 00043
67106   00 00000 00045
67107   11 50404 65025
67110   54 20000 00153
67111   37 77777 77777
67112   20 00000 00000
67113   00 00014 00000
67114   00 00014 00014
67115   00 00000 00021
67116   00 00000 00004
67117   00 00012 00000
```

PX 71900-9-(135)

THE RAMO-WOOLDRIDGE CORPORATION
Los Angeles 45, California

STATED POINT CARD OUTPUT

Specifications

Identification Tag:	CPO-2
Type:	Subroutine
Assembly Routine Spec:	SUB 50953 11800 (Note: 154 consecutive cells required for execution)
Storage:	118 instructions plus 36 cells which must be reserved by the user of this subroutine, immediately following the subroutine. 154 words total program storage 6 words temporary storage pool used, addresses 27b through 34b The constant pool is used by this routine.
Program Entrance:	Address SUB00 (Note: non-conventional entrance and exit)
Program Exit:	Address SUB00
Alarm Exit:	The alarm exit is not used by this routine, but if a number to be punched is too large (see programming instructions) a divide fault and an I0 fault will occur.
Machine Time:	0.5 seconds per card. 6 numbers per card.
Mode of Operation:	Stated point.

Coded by:	M. Perry	April, 1956
Code Checked by:	M. Perry	April, 1956
Machine Checked by:	M. Perry	April, 1956
Approved by:	W. F. Bauer	April, 1956

Description

This subroutine punches six stated-point, rounded decimal numbers per card. Each card also contains an integer to identify the deck, and a one digit card number.

Programming Instructions

The subroutine must be stored in high speed storage. Assume that it is located at address SUB00. To enter the subroutine use:

RJ SUB00 SUB00

followed by a control word of the form

NN L M

The operation part of the control word, NN, is two octal digits which specify the number of numbers to be punched.

The u-address part of the control word, L, is the address of the first cell of a list of parameter words, the make-up of which is described below. This list of parameter words starting at cell L must be in high speed storage.

The v-address part of the control word, M, is the address of an identification integer, which must be in high speed storage. The identification integer located in cell M must be scaled 2^0 and must be less than 10^7. If (M) $\geqslant 10^7$ a divide check will occur.

The list of parameter words occupies NN consecutive cells in high speed storage starting with cell L. Each parameter word, one for each number to be punched, contains four pieces of information in the form

SS P OIIFF

The operation part of the parameter word, SS, gives the binary scaling of the number to be punched. This scaling information must be expressed in two octal digits. If SS is greater than 43b (35 decimal) then II must be specified as 00.

The u-address part of the parameter word, P, is the address of the number to be punched, which must be in high speed storage.

Of the five octal digits which make up the v-address part of the parameter word the first must be zero; the next two, II, specify the number of decimal digits to be punched in the integer part of

the number. If SS is greater than 43b (35 decimal) then II must be 00.

The last two octal digits, FF, specify the number of decimal digits to be punched in the fractional part of the number.

The total number of decimal digits to be punched must not exceed ten (decimal). (i.e., II + FF $\leqslant$ 12b).

If the number to be punched is too large to be expressed in the number of digits specified a divide fault and an I0 fault will occur.

Note: These parameter words are similar to those used by HT0-0.

OUTPUT CARD FORM

Each card punched by this subroutine has 8 fields as follows:

Field	Columns	Contents
1	1 - 12	1st number
2	13 - 24	2nd "
3	25 - 36	3rd "
4	37 - 48	4th "
5	49 - 60	5th "
6	61 - 72	6th "
7	73 - 79	Identifying Integer
8	80	card number

In each of the six number fields the 12th or right most column contains the sign of the number. The 11th column contains the least significant digit to be punched. The remaining digits and the decimal point occupy succeeding columns to the left as far as required. Zeros to the left of the most significant digit or decimal point are suppressed.

Field 7 contains the 7 digits of the identifying integer with no decimal point.

Field 8 contains a single decimal digit card number, (modulo 10). The first card produced by each entry into the routine contains a 1, the second a 2, and so on up to 9, 0, and 1 again.

TIMING

As noted above all cells referred to by this routine (namely the control word, the identification integer, the parameter list, and all numbers to be punched) should be in high speed storage. Successive entries at intervals of 9 ms or less may then be made without interrupting the card cycle.

```
D          S0P00 50953                                  65311  00 00000 00000
D          S1P00 50974                                  65336  00 00000 00000
D          S2P00 51026                                  65422  00 00000 00000
D          S3P00 51039                                  65437  00 00000 00000
D          S4P00 51047                                  65447  00 00000 00000
D          0CP00 00013                                  00015  00 00000 00000
D          00P00 01024                                  02000  00 00000 00000
D          01P00 01045                                  02025  00 00000 00000
D          02P00 01097                                  02111  00 00000 00000
D          03P00 01110                                  02126  00 00000 00000
D          04P00 01118                                  02136  00 00000 00000
D          00T00 00023                                  00027  00 00000 00000
D          07P00 01142                                  02166  00 00000 00000
D          08P00 01154                                  02202  00 00000 00000
D          09P00 01166                                  02216  00 00000 00000
S0P00   MP 0CP03 00000          ENTRY                   65311  71 00020 00000
S0P01   TU A0000 01P03          LOC PARAM               65312  15 20000 02030
S0P02   TV A0000 04P23          LOC IDENT               65313  16 20000 02165
S0P03   SP A0000 00005          NO WORDS                65314  31 20000 00005
S0P04   TP B0000 00T00                                  65315  11 30000 00027
S0P05   TV 00P00 01P44          SET FINAL               65316  16 02000 02101
S0P06   RA 01P44 0CP03          EXIT                    65317  21 02101 00020
S0P07   TP 0CP03 00T01          CARD NUMBER             65320  11 00020 00030
S0P08   IJ 00T00 00P09          REDUCE N                65321  41 00027 02011
S0P09   TP 00P20 00T02          U FOR FST WD            65322  11 02024 00031
S0P10   EF 00000 01P25                                  65323  17 00000 02056
S0P11   RP 20036 00P13                                  65324  75 20044 02015
S0P12   CC 07P00 A0000                                  65325  27 02166 20000
S0P13   TU 00P19 02P12                                  65326  15 02023 02125
S0P14   RJ 01P01 01P02          FIRST 3                 65327  37 02026 02027
S0P15   TU 01P01 02P12                                  65330  15 02026 02125
S0P16   RJ 01P01 01P00          SECOND 3                65331  37 02026 02025
S0P17   RJ 04P17 04P00          IDENT PUNCH             65332  37 02157 02136
S0P18   RA 00T01 0CP03          BUMP IDENT              65333  21 00030 00020
S0P19   45 07P02 00P09 BRR                              65334  45 02170 02011
S0P20   40 00000 00000 B                                65335  40 00000 00000
S1P00   QJ 01P01 01P02          SET 1                   65336  44 02026 02027
S1P01   45 08P02 00000 BRB                              65337  45 02204 00000
S1P02   LQ 00T02 00026          EXIT                    65340  55 00031 00032
S1P03   11 00000 10112 B        PROC PARA               65341  11 00000 10112
S1P04   RA 01P03 0CP02          ADV PAR LOC             65342  21 02030 00017
S1P05   SN Q0000 00042                                  65343  33 10000 00052
S1P06   AT 01P45 01P28          SHIFT INST              65344  35 02102 02061
S1P07   QT 0CP01 00T04          FD                      65345  51 00016 00033
S1P08   TU Q0000 01P26          LOC                     65346  15 10000 02057
S1P09   LQ Q0000 00030                                  65347  55 10000 00036
S1P10   QT 0CP01 00T03          ID                      65350  51 00016 00032
S1P11   ZJ 01P12 01P18                                  65351  47 02041 02047
S1P12   SS 0CP03 00015                                  65352  34 00020 00017
S1P13   TU A0000 01P16                                  65353  15 20000 02045
```

PX 71900-9-(136)

```
S1P14   TP 01P48 01P30          ID NON ZERO      65354  11 02105 02063
S1P15   TP 01P46 A0000                           65355  11 02103 20000
S1P16   RP 00000 01P19          ID LESS 1        65356  75 00000 02050
S1P17   MP A0000 01P46                           65357  71 20000 02103
S1P18   TP 01P49 01P30          ID ZERO          65360  11 02106 02063
S1P19   TP A0000 02P08          DIVISOR          65361  11 20000 02121
S1P20   SP 00T04 00015          FD               65362  31 00033 00017
S1P21   TU A0000 01P23                           65363  15 20000 02054
S1P22   SP 00P20 00034          ONE HALF  70     65364  31 02024 00042
S1P23   RP 00000 01P25          FD               65365  75 00000 02056
S1P24   MP B0000 01P47                           65366  71 30000 02104
S1P25   11 30000 10112 B                         65367  11 30000 10112
S1P26   TP 00000 00T05          PROCURE DATA     65370  11 00000 00034
S1P27   TM 00T05 A0000                           65371  12 00034 20000
S1P28   54 20000 00153 B        SHIFT            65372  54 20000 00153
S1P29   AT Q0000 01P28          ROUNDED   35     65373  35 10000 02061
S1P30   DV 02P08 01P28                           65374  73 02121 02061
S1P31   LQ 01P28 00001                           65375  55 02061 00001
S1P32   TV 00T03 01P34          ID               65376  16 00032 02067
S1P33   RA 01P34 00T04          FD               65377  21 02067 00033
S1P34   LQ 00T02 00000          POSITION 1       65400  55 00031 00000
S1P35   TV 01P50 02P04          SET ZERO SUP     65401  16 02107 02115
S1P36   RJ 02P11 02P10          INTEGRL DIG      65402  37 02124 02123
S1P37   RJ 02P11 03P00          DEC PT           65403  37 02124 02126
S1P38   TP 00T04 00T03          FD               65404  11 00033 00032
S1P39   RJ 02P11 02P10          FRACT DIG        65405  37 02124 02123
S1P40   TJ 00T05 01P50                           65406  42 00034 02107
S1P41   RJ 02P11 02P05                           65407  37 02124 02116
S1P42   IJ 00T00 01P00                           65410  41 00027 02025
S1P43   RJ 04P17 04P00          IDENT PUNCH      65411  37 02157 02136
S1P44   45 09P02 00000 BRB                       65412  45 02220 00000
S1P45   54 20000 00153 B                         65413  54 20000 00153
S1P46   00 00000 00012 B        TEN              65414  00 00000 00012
S1P47   01 00000 00000 -01  35 ONE TENTH         65415  03 14631 46315
S1P48   DV 02P08 01P28                           65416  73 02121 02061
S1P49   MJ 00008 01P31                           65417  45 00010 02064
S1P50   RJ 02P11 02P09                           65420  37 02124 02122
S1P51   MJ 00000 01P42                           65421  45 00000 02077
S2P00   SP 01P28 00002                           65422  31 02061 00002
S2P01   SA 01P28 00001                           65423  32 02061 00001
S2P02   TP A0000 01P28                           65424  11 20000 02061
S2P03   SS A0000 00051                           65425  34 20000 00063
S2P04   ZJ 02P06 00000                           65426  47 02117 00000
S2P05   SN 0CP02 00000                           65427  33 00017 00000
S2P06   TV 03P06 02P04                           65430  16 02134 02115
S2P07   AT 02P12 02P08                           65431  35 02125 02121
S2P08   RA 07P02 00T02                           65432  21 02170 00031
S2P09   LQ 00T02 00035                           65433  55 00031 00043
```

PX 71900-9-(136)

```
S2P10   IJ 00103 02P00                          65434  41 00032 02111
S2P11   MJ 00000 00000                          65435  45 00000 00000
S2P12   RA 07P01 00T02                          65436  21 02167 00031
S3P00   SN 00015 00001                          65437  33 00017 00001
S3P01   AT 02P12 03P03                          65440  35 02125 02131
S3P02   AT 03P07 03P04                          65441  35 02135 02132
S3P03   RA 00000 00000                          65442  21 00000 00000
S3P04   RA 00000 00000                          65443  21 00000 00000
S3P05   SP 00015 00003                          65444  31 00017 00003
S3P06   MJ 00000 02P06                          65445  45 00000 02117
S3P07   00 00005 00000                          65446  00 00005 00000
S4P00   TP 0CP03 00T02  SET MASK                65447  11 00020 00031
S4P01   LQ 04P23 10015  PAR WD TO Q              65450  55 02165 10017
S4P02   TP 01P44 01P37  SET EXIT                65451  11 02101 02072
S4P03   TU 01P44 02P12  SET IMAGE               65452  15 02101 02125
S4P04   RJ 01P37 01P05  GEN IMAGE               65453  37 02072 02032
S4P05   TP 04P19 01P37  RESET                   65454  11 02161 02072
S4P06   SP 00T01 00015  GEN IMAGE              65455  31 00030 00017
S4P07   RJ 02P11 02P07   CARD NO                65456  37 02124 02120
S4P08   TP 04P18 00000                          65457  11 02160 10000
S4P09   RP 30003 04P11  SET TO WRITE            65460  75 30003 02151
S4P10   TP 04P20 04P11   9 ROW                  65461  11 02162 02151
S4P11   EW 00000 09P10  WRITE                   65462  77 00000 02230
S4P12   EW 10000 07P10                          65463  77 10000 02200
S4P13   EW 10000 08P10                          65464  77 10000 02214
S4P14   RP 20003 04P16                          65465  75 20003 02156
S4P15   RS 04P11 0CP03                          65466  23 02151 00020
S4P16   IJ 00000 04P11  LOOP                    65467  41 10000 02151
S4P17   MJ 00000 00000                          65470  45 00000 00000
S4P18   00 00000 00011                          65471  00 00000 00013
S4P19   RJ 02P11 03P00                          65472  37 02124 02126
S4P20   EW 00000 09P11                          65473  77 00000 02231
S4P21   EW 10000 07P11                          65474  77 10000 02201
S4P22   EW 10000 08P11                          65475  77 10000 02215
S4P23   00 70000 00000                          65476  00 70000 00000
```

PX 71900-9-(136)

THE RAMO-WOOLDRIDGE CORPORATION
Los Angeles 45, California

OCTAL CARD DUMP

Specifications

Identification Tag:	MDP-4
Type:	Service routine (with subroutine entrance)
Special Storage:	The constant and temporary storage pools are not used by this routine.
Service Entrance:	Address 40015b
Program Entrance:	Address 40015b
Program Exit:	Address 40020b
Alarm Exit:	The alarm exit is not used by this routine.
Machine Time:	2.7 seconds plus 0.5 seconds per card maximum machine time
	6 words per card

Coded by:	C. Koos	April, 1956
Machine Checked by:	C. Koos	April, 1956
Approved by:	W. F. Bauer	April, 1956

Description

This routine will dump the contents of a group of consecutive ES or MD storage cells onto cards. Each card will contain six consecutive octal words and the address of the cell containing the first word on the card. The following card columns are used:

Columns 1 - 12	first word
Columns 13 - 24	second word
Columns 25 - 36	third word
Columns 37 - 48	fourth word
Columns 49 - 60	fifth word
Columns 61 - 72	sixth word
Columns 76 - 80	address of first word

Any card for which all six words consist of 36 binary zeros is omitted and the next card produced carries a punch in the 12 row of column 74. The first and last cards of every dump will be produced even if they contain all zeros. Zeros will not be punched at any time in columns 37 - 72. If the last card reaches the end of ES or of MD before it is filled, the remaining columns will be completed by using MD words, beginning with address 40000.

Each card carries an identifying punch in the 12 row of column 73.

This routine bootstraps itself into ES to operate and then positions cards. At its conclusion it restores the machine to its original state, and clears the bull.

OPERATING INSTRUCTIONS (to be followed when the routine is used as a service routine).

1. Put the computer in test mode, high speed (this step is unnecessary for a dump of all ES only).

2. Set PAK to 40015b and start.

3. Computation will halt with an MSO instruction and Q will contain all zeros.

4. Manually insert the parameter word into Q.

   a. a parameter word of all zeros will dump ES.

   b. in all other cases, place the address of the first word to be dumped in Q_u and the address of the last word to be dumped in Q_v

   Notes: If the starting address is an ES address and the last address is either illegal or a drum address, the routine will dump up to the end of ES and then exit. An immediate exit will occur and no cards will be punched

if the starting address is illegal or exceeds the last address.

5. The machine will halt with an MSO instruction when the dump is completed and the machine has been restored to its original state.

6. If another dump is required, it is necessary only to press the start button again to return to step 3 above.

7. If the operator wishes to stop a dump at any time after step 3 above, he needs only to make a forced stop, master clear, and MD start with PAK set to 40040b. The machine will then be restored to its original state and computation will halt with the same MSO instruction mentioned in step 5.

PROGRAMMING INSTRUCTIONS (to be followed when the routine is used as a subroutine).

1. Enter the routine with the instruction 37 40020 40015b. The word in your program immediately following the RJ instruction must contain the parameter word (as described in step 4 of "Operating Instructions" above). If the RJ instruction is given at address n the parameter word will be at address n + 1 and at the conclusion of the dump control will be returned to the instruction address n + 2.

C H I P

An Interpretive Subroutine for Packed Floating Point Operands For the ERA 1103 Computer

Programmed by:
L. Fall
P. Malone

PX 71900-9-(138)

Programmers at Wright Field have felt the need for a compact floating point interpretive coding system. This interpretive system should include basic floating point arithmetic, elementary transcendental functions and floating decimal Flex-coded print or punch output.

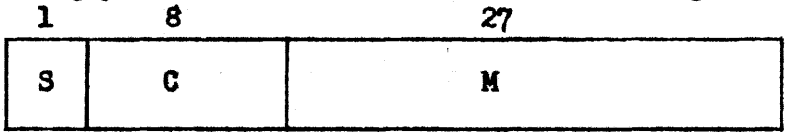
A more convenient packed floating number definition is desirable. Consequently, the packed floating number definition of the 1103AF was chosen.

A floating point "Chip" number N must satisfy one of the following conditions:

1) $N = 0$

2) $2^{-129} \leq |N| < 2^{127}$

The floating point word structure has the following form:



where S - u_{35} - sign

C - u_{34}, --- U_{27} - characteristic

M - u_{26}, --- U_0 - mantissa

S and M denote the 1's complement representation of $x.2^{27}$, while C is the representation of $\lfloor y + 128 \rfloor$ scaled 27 or when S = 1, C is the 1's complement form of $\lfloor y + 128 \rfloor$ scaled 27.

For N = 0, S = C = M = all 0's, or all 1's.

For $2^{-129} \leq N < 2^{127}$, N is represented in the form $x.2^y$, where $\frac{1}{2} \leq x < 1$ and $-128 \leq y \leq 127$. Then, S = 0, M = x (scaled 27), and C = $\lfloor y + 128 \rfloor$ (scaled 27).

For $-2^{127} < N \leq -2^{-129}$, N is represented by the one's complement of $|N|$.

00 Repeat i | n | w

01 R + u' ⟶ R, A, Q

02 R − u' ⟶ R, A, Q

03 R · u' ⟶ R, A, Q

04 R ÷ u' ⟶ R, A, Q

05 R + R·u' ⟶ R, A, Q

06 u' + R·u' ⟶ R, A, Q

07 (R + u') · R → R, A, Q

10 Repeat i | n | w

11 u' + v' ⟶ R, A, Q

12 u' − v' ⟶ R, A, Q

13 u' · v' ⟶ R, A, Q

14 u' ÷ v' ⟶ R, A, Q

15 R + u' · v' → R, A, Q

16 u' + R · v' → R, A, Q

17 (u' + v') · R → R, A, Q

20 Repeat i | n | w

21 R + u' → v', R, A.

22 R − u' → v', R, A.

23 R · u' → v', R, A.

24 R ÷ u' → v', R, A.

25 R + Ru' → v', R, A.

26 u' + R·u' → v', R, A.

27 (R + u') · R → v', R, A.

30 Repeat i | n | w

31 u' + v' ⟶ v', R, A.

32 u' − v' ⟶ v', R, A.

33 u' · v' ⟶ v', R, A.

34 u' ÷ v' ⟶ v', R, A.

35 R + u' · v' → v', R, A.

36 u' + R·v' ⟶ v', R, A.

37 (u' + v') · R → v', R, A.

40 *cos u' ⟶ v', R, A.

41 *sin u' ⟶ v', R, A.

42 $\sqrt{u'}$ ⟶ v', R, A.

43 $e^{u'}$ ⟶ v', R, A.

44 ln u' ⟶ v', R, A.

45 *$\tan^{-1}$ u' ⟶ v', R, A.

46 Print u' (car ret if v' = o)

47 Punch u' (car ret if v' = o)

*Radians

PX 71900-9-138

The CHIP instructions are in the following form:

14	OP	u'	v'

in which OP is a two-octal-digit pseudo-code and u' and v' are four-octal-digit address references. 14 occupies $u_{35}, \ldots u_{30}$; OP occupies $u_{29}, \ldots u_{24}$; u' occupies $u_{23}, \ldots u_{12}$; and v' occupies $u_{11}, \ldots u_{0}$. Because of the four-octal-digit limit on address references, operands must be located in HSS (High-speed storage).

HSS address 00005 is designated as R. After every interpretive instruction the normalized, rounded, packed result is left in R and the double extension of R is left in A.

The repeat order ,00, functions in a manner analogous to regular 1103 operation. It is coded in the form:

14	00	j	n	w

, j occupying U_{23}, U_{22}, U_{21}. W denotes the address from which the next instruction will be taken after the termination of the repeat. Repeat orders, 10, 20, 30, operate in exactly the same way as above.

It should be noted that addresses 00003 thru 00016 are used as temporary storage. Using 00005 as a u' or v' address will yield the correct result except in the following cases:

14	06	0005	- - - -	gives	$2R^2$
14	16	0005	v'	"	2Rv'
14	26	0005	v'	"	$2R^2$
14	36	0005	v'	"	2Rv'

The alarm exit of "Chip" is at 00002. The following "Chip" alarm entry has been added to the alarm print routine (CV-3) which places the packed operands

of the current instruction in A_l and A_r :

75750	37	00224	00224
1	31	00015	00044
2	27	20000	00016
3	16	00000	75700
4	45	00000	75701

For those operations involving only one operand, the contents of u' will be placed in Ar when an alarm occurs. If an alarm occurs during a repeated instruction at address y, the address printed out will be (W-1) which may or may not be y.

An alarm will occur if the result of an interpretive operation exceeds the limits of the "Chip" number definition. This means that a result too small in absolute magnitude will also give an alarm. By changing address 00213 from the 46 00002 00216 to 46 00214 00216 all results such that $0 < |\text{Result}| < 2^{-129}$ will be replaced by zero without an alarm.

Furthermore, an alarm occurs when a division by zero or by an unnormalized number is attempted; when the square root of a negative number is attempted; if the absolute value of the argument in the sine-cosine routine exceeds 2^{18}., if the argument in $e^{u'}$ is less than -2^{18} or greater than $\ln 2^{127}$; or if in the logarithm routine, the argument is equal to or less than zero.

CODE		FUNCTION	LIMITS	TIME	INTERVAL TESTED	ERROR	METHOD
40, 41	*	Cosine, Sine	$\|u'\| < 2^{18}$	7.8ms	0(.05) 20	$\|E \text{ max.}\| < 10^{-7}$	Rand Poly - Sheet 16
42		Square Root	$0 \leq u' < 2^{127}$	7.5ms	0(100) 12,500	$\|Er\| < 2^{-27}$	Newton
43		Exponential	$-2^{18} < u' < \ln 2^{127}$	14.9ms	-2.5 (.02)2.5 -77(1) 77	$\|Er\| < 2^{-27}$ $\|Er\| < 2^{-27}$	Power Series
44		Natural Log	$0 < u' < 2^{127}$	7.1ms	.02(.02) 5.0	$\|E \text{ max}\| < 10^{-07}$	Rand Poly - Sheet 56
45	*	Arc Tangent	$\|u'\| < 2^{127}$	8.0ms	-20(.1) 20	$\|E \text{ max}\| < 10^{-7}$	Rand Poly - Sheet 13

* Radians

Where E max is maximum error
Er is maximum relative error

Average Times for Other Orders

X 1	Addition	3.2 ms.
X 2	Subtraction	3.3 ms.
X 3	Multiplication	3.1 ms.
X 4	Division	3.6 ms.
X 5		5.1 ms.
X 6	Multiple Orders	5.1 ms.
X 7		5.1 ms.
4 6	Print	2^{sec}/word
4 7	Punch	$.27^{sec}$/word

Printed or Punched Output - Use of the print (46) or punch (47) orders causes the packed floating point number at address u' to be printed or punched in floating decimal form. The format for such a number contains fourteen characters in the form:

$\overline{sp}$	X	.	X	X	X	X	X	X	X	$\overline{sp}$	Y	Y	SP
1	2	3	4	5	6	7	8	9	10	11	12	13	14

Characters 1 thru 10 print out the decimal mantissa and characters 11 thru 13 print out the power of ten by which the mantissa is to be multiplied. Every such number is followed by a space.

There is also the option available to the programmer for a carriage return before printing or punching. This is accomplished by putting zero's in the v' portion of the print or punch order. Used with a repeat order which has j=3, it is possible to print or punch several numbers in consecutive storage addresses on one line using two lines of coding.

```
B     14 00 3 005   B+2
B+1   14 47 0 700   0000
B+2   continuance of program
```

Using these orders, a programmer has at his disposal a wide variety of possible formats for output.

It should be noted that "Chip" is <u>not</u> a standard subroutine. The entries, temporary storage, constant pool, basic arithmetic, and function routines occupy in that order the first 700_8 HSS addresses.

While the choice of pseudo-codes available in the interpretive repertoire is perhaps not as complete as some might desire, it is felt that considerable programming flexibility is achieved while compactness is retained.

00000	45 00000(30000)	Normal Exit
00001	45 00000 00100	Entry
00002	45 00000 75750	Alarm Exit
00003	00 00000 00000	Mantissa of U Operand
		TEMPORARY STORAGE
00004	00 00000 00000	Characteristic of U Operand
00005	00 00000 00000	Chip Accumulator "R," also Mantissa of V Operand
00006	00 00000 00000	Characteristic of V Operand
00007	00 00000 00000	U^1 Address Stored in U-Portion
00010	00 00000 00000	V^1 Address Stored in U & V-Portions
00011	00 00000 00000	Repeat Counter
00012	00 00000 00000	j Counter
00013	00 00000 00000	Temporary Storage for R
00014	00 00000 00000	Current Instruction
00015	00 00000 00000	Packed U-Operand
00016	00 00000 00000	Packed V-Operand
00017	00 04000 00000	.5 Scaled 27
		CHIP CONSTANTS
00020	20 14000 00000	Chip 1
00021	02 00000 00000	1 Scaled 31
00022	13 05620 57737	ln 2 Scaled 34
00023	54 00005 24110	Scaling Constant
00024	00 00000 00030	24_{10}
00025	14 44176 65211	π Scaled 32
00026	06 22077 32504	$\pi/2$ Scaled 32
00027	00 00000 00100	64_{10}

PX 71900-9-(138)

00030	00 00000(30000)	Scale Factor Storage
00031	00 00000 00200	128_{10}
00032	00 00000 00223	147_{10}
00033	40 07777 77777	Mantissa Mask
00034	37 70000 00000	Characteristic Mask
00035	00 00000 00777	Three-Octal-Digit Extractor
00036	00 00000 00206	134_{10}
00037	00 00000 00045	37_{10}

CONSTANT POOL

00040	00 00000 00000	Zero
00041	00 00000 00002	2 and "Color Shift"
00042	61 00000 00045	Print and "Car. Return"
00043	00 00000 00003	3
00044	00 00000 00004	4
00045	00 00000 00037	Flex Code 0
00046	00 00000 00052	Flex Code 1
00047	00 00000 00074	Flex Code 2
00050	00 00000 00070	Flex Code 3
00051	00 00000 00064	Flex Code 4
00052	00 00000 00062	Flex Code 5
00053	00 00000 00066	Flex Code 6
00054	00 00000 00072	Flex Code 7
00055	00 00000 00060	Flex Code 8
00056	00 00000 00033	Flex Code 9
00057	00 00000 00013	11_{10}

PX 71900-9-(138)

00060	00 00000 00012	10_{10}
00061	00 00000 00056	"_"
00062	31 10375 52421	$\pi/4$ Scaled 35; $\pi/2$ Scaled 34
00063	31 46314 63146	0.1_{10} Scaled 38
00064	00 00000 00077	Six-Bit Extractor
00065	21 67643 24177	Degrees to Radians Scaled 40
00066	20 00000 00000	.5 Scaled 35
00067	00 00000 00007	Octal Digit Extractor
00070	37 77777 77777	$2^{35}-1$
00071	00 77777 00000	U Extractor
00072	00 00000 77777	V Extractor
00073	00 00001 00000	U Advance
00074	00 00000 00001	V Advance
00075	00 00001 00001	U and V Advance
00076	00 07777 07777	Four-Octal Digit U and V Extractors
00077	00 00000 00110	72_{10}

00100	11 00040 00011	Set Repeat Ctr. to Zero
		EXPANSION
00101	31 00000 00000	
00102	34 00074 00017	
00103	15 20000 00104	
00104	(11 (30000) 10000)	Transmit Current Instruction to Q
00105	11 10000 00014	Store Current Instruction in 00014
00106	51 00076 00010	Extract V^1
00107	31 20000 00017	

```
00110   15 20000 00010   v^1 · 2^15 + v^1 Stored in 00010
00111   55 10000 00003
00112   51 00076 00007   u^1 · 2^15 Stored in U-Portion of 00007
-------------------------------------------------------------------
00113   55 00014 10006   Shift Current Instruction Six Places in Q
00114   16 00010 00225   Set up:  Store Result at v^1
00115   15 00117 00147   Set up:  R as U Operand
00116   15 00007 00157   Set up:  u^1 as V Operand
00117   11 00005 00013   Store Previous Result in 00013
00120   44 00135 00121   Test PC5          PC = Pseudo-Code
00121   44 00123 00122   Test PC4
00122   16 00104 00225   Set up:  Store Result at Q
00123   44 00124 00126   Test PC3
00124   15 00007 00147   Set up:  u^1 as U Operand   DECODING
00125   15 00010 00157   Set up:  v^1 as V Operand
00126   44 00132 00127   Test PC2
00127   44 00131 00130   Test PC1
00130   44 00261 00237   X1  X0    Entries for X = 0, 1, 2, 3.
00131   44 00147 00256   X3  X2
00132   44 00134 00133   Test PC1
00133   44 00307 00246   X5  X4    Entries for X = 0, 1, 2, 3.
00134   44 00317 00313   X7  X6
00135   37 00167 00157   Unpack for Functions
00136   11 00016 20000   Contents of u^1 into A
00137   55 00014 10007   Shift Current Instruction Seven Places in Q
```

PX 71900-9-(138)

00140	44 00002 00141	Alarm for Undefined Pseudo-Codes
00141	44 00002 00142	Alarm for Undefined Pseudo-Codes
00142	44 00145 00143	Test PC_2
00143	44 00144 00343	Sine-Cosine Entry
00144	44 00422 00323	Entry for Exponential or Square Root
00145	44 00557 00146	Print/Punch Entry
00146	44 00505 00453	Entry for Arc Tan or Logaritm
00147	11(30000)00015	Store U Operand
		UNPACK
00150	11 00015 10000	
00151	51 00033 00003	Store U Mantissa
00152	51 00034 00004	Store U Characteristic
00153	44 00154 00156	
00154	27 00003 00034	
00155	27 00004 00034	
00156	55 00004 00011	
00157	11(30000)00016	Store V Operand
00160	11 00016 10000	
00161	51 00033 00005	Store V Mantissa
00162	51 00034 00006	Store V Characteristic
00163	44 00164 00166	
00164	27 00005 00034	
00165	27 00006 00034	
00166	55 00006 00011	
00167	37 00167(00170)	

00170	21 00006 00004	MULTIPLICATION
00171	36 00032 00006	
00172	71 00003 00005	
00173	47 00174 00220	NORMALIZE, ROUND, PACK
00174	11 00031 00005	
00175	46 00176 00177	
00176	13 00031 00005	
00177	74 20000 00030	
00200	11 20000 10000	
00201	21 10000 00005	
00202	43 10000 00205	
00203	21 00006 00074	
00204	55 00005 10033	
00205	54 10000 00100	
00206	11 00030 20000	
00207	42 00037 00211	
00210	36 00077 20000	
00211	35 00006 20000	
00212	54 20000 00033	
00213	46 00002 00216	NOTE: (00002) is Alarm Exit for $0 < \|Result\| < 2^{-129}$ (00214) Replaces Result
00214	11 00040 20000	By Zero With No Alarm.
00215	45 00000 00220	
00216	42 00070 00221	
00217	45 00000 00002	Alarm, Characteristic Too Large

PX 71900-9-(138)

00220	11 20000 10000		
00221	52 00033 00005		
00222	44 00223 00224		
00223	27 00005 00034		
00224	37 00224 (00225)		
00225	11 20000 (30000)	Store Result	TERMINATION
00226	41 00011 00231	Repeat Instruction?	
00227	11 00005 20000	Result to A	
00230	45 00000 00000	Jump to Exit	
00231	55 00012 10001	j to Q	REPEAT MODIFICATION
00232	44 00233 00234		
00233	21 00007 00075	Advance u^1	
00234	44 00235 00236		
00235	21 00010 00075	Advance v^1	
00236	45 00000 00113		
00237	11 10000 00012	Store j	SET UP REPEAT
00240	16 00010 00000	w to V-Portion of F_1	
00241	21 00104 00073		
00242	55 00014 10030		
00243	51 00035 00011	Store n in 00011	
00244	41 00011 00104	Store n - 1 in 00011; Jump to Next Instruction	
00245	45 00000 00227	Exit if n = 0	
00246	37 00167 00147	Unpack	DIVIDE
00247	12 00005 20000		

```
00250   42 00017 00002   Alarm if Divisor Unnormalized or Zero
00251   23 00004 00006
00252   35 00036 00006
00253   54 00003 20035
00254   73 00005 20000
00255   45 00000 00173   Jump to Pack
-------------------------------------------------------------
00256   37 00167 00147   Unpack                    SUBTRACT
00257   13 00005 00005   Negate V Operand
00260   45 00000 00262   Jump to Add
-------------------------------------------------------------
00261   37 00167 00147   Unpack                    ADD
00262   11 00004 20000
00263   36 00006 20000
00264   46 00275 00265
00265   42 00037 00271
00266   11 00004 00006
00267   54 00003 20010
00270   45 00000 00306
00271   16 20000 00272
00272   54 00003 (30000)
00273   35 00005 20000
00274   45 00000 00305
00275   13 20000 20000
00276   42 00037 00301
00277   54 00005 20010
```

PX 71900-9-(138)

00300	45 00000 00306		
00301	16 20000 00302		
00302	54 00005 (30000)		
00303	35 00003 20000		
00304	11 00004 00006		
00305	54 20000 20010		
00306	45 00000 00173	Jump to Pack	
00307	37 00224 00147	U·V	$R \leftarrow u^1 \cdot v^1$ or $R \leftarrow R \cdot u^1$
00310	15 00312 00147	Set up: Previous Result as U	
00311	15 00164 00157	Set up: R as V	
00312	45 00013 00261	Jump to Add	
00313	15 00312 00147	Set up: Previous Result as U	$u^1 \leftarrow R \cdot u^1$ or $u^1 \leftarrow R \cdot v^1$
00314	37 00224 00147	U·V	
00315	15 00007 00147	Set up: u^1 as U	
00316	45 00000 00311	Jump to (U ← R)	
00317	37 00224 00261	U ← V	$(u^1 \leftarrow v^1) \cdot R$ $(R \leftarrow u^1) \cdot R$
00320	15 00164 00147	Set up: R as U	
00321	15 00312 00157	Set up: Previous Result as V	
00322	45 00000 00147	Jump to Multiply	
00323	46 00002 00324	Alarm if Argument Negative	$\sqrt{u^1}$
00324	47 00325 00173	If Zero, Jump to Exit	
00325	31 00006 00107		
00326	46 00327 00330		
00327	55 00005 00001		

PX 71900-9-(138)

```
00330    35 00027 00006
00331    11 00066 10000
00332    11 10000 00003
00333    31 00005 00051
00334    73 00003 20000
00335    32 00003 00107
00336    11 20000 10000
00337    23 20000 00003
00340    47 00332 00341
00341    31 10000 00001
00342    45 00000 00173    Jump to Pack
```

```
00343    11 00040 00003    Store Zero in 00003 for Sine
00344    44 00346 00345    Test PC0                          SINE-COSINE
00345    11 00026 00003    Store π/2  in 00003 for Cosine
00346    23 00006 00032
00347    46 00350 00002    Alarm if |u1| ≥ 2^18 = 262,144.
00350    35 00055 20000
00351    46 00352 00353
00352    11 00040 00005    Replace u1 By Zero
00353    36 00024 10000
00354    35 00023 00355
00355    (00 00000 00000)  u1 Into A Scaled 32
00356    44 00357 00360
00357    11 20000 20000    Zero Into AL if U1 Scaled "Down"
```

```
00360    35 00003 20000
00361    73 00062 10000    0 ≤ A_R < 2π Scaled 32
00362    11 00066 00003    Store Sign in 00003
00363    42 00026 00367
00364    36 00025 20000
00365    55 00003 00001    Shift Sign
00366    45 00420 00363
00367    54 20000 00042
00370    73 00026 00005    X Into 00005 Scaled 34
00371    71 10000 10000
00372    54 20000 00046
00373    11 20000 00006    x^2 Into 00006 Scaled 34
00374    11 00421 00004    C_9 Into P_i
00375    15 00366 00401    Set u Address of 00461
00376    11 00043 00013    Set Index
00377    71 00006 00004    x^2 . P_i
00400    54 20000 00046
00401    35 30415 00004    P_i + 1
00402    23 00401 00073
00403    41 00013 00377
00404    71 00005 00004    x . P
00405    54 20000 00046
00406    11 20000 00005    Store result in 00005
00407    11 00003 10000
```

00410	44 00411 00412	Examine Sign
00411	13 20000 00005	
00412	11 00031 00006	
00413	54 00005 20001	
00414	45 00000 00173	Jump to Pack
00415	31 10375 52202	C_1 Rand Coefficients Scaled 34
00416	65 52420 76452	C_3 Rand Coefficients Scaled 34
00417	01 21464 25731	C_5 Rand Coefficients Scaled 34
00420	77 73155 46346	C_7 Rand Coefficients Scaled 34
00421	00 00117 32757	C_9 Rand Coefficients Scaled 34

00422	23 00006 00032	EXPONENTIAL
00423	46 00424 00002	Alarm
00424	35 00055 20000	
00425	46 00426 00427	
00426	11 00040 00005	Zero to 00005
00427	36 00451 10000	
00430	35 00023 00431	
00431	(00 00000 00000)	u^1 Into A Scaled 34
00432	44 00433 00434	
00433	11 20000 20000	Zero to A_L if u^1 Scaled "Down"
00434	73 00022 00006	
00435	11 20000 00005	x Into 5
00436	11 00452 00003	11_{10} Into 00003 Scaled 31
00437	11 00021 00004	1 Into 00004 Scaled 31

PX 71900-9-(138)

00440	71 00005 00004	$x \cdot P_i$
00441	73 00003 20000	
00442	32 00066 00105	
00443	11 20000 00004	P_{i+1} Into 00004 Scaled 31
00444	23 00003 00021	
00445	47 00440 00446	
00446	21 00006 00031	Characteristic Into 00006
00447	31 00004 00004	e^x Into A Scaled 35
00450	45 00000 00173	Go to Pack
00451	00 00000 00026	22_{10}
00452	26 00000 00000	11_{10} Scaled 31
00453	23 00006 00473	P - 1
00454	54 00005 20010	2q Into A Scaled 34
00455	36 00066 00005	2q-1 Into 00005
00456	46 00002 00457	Alarm
00457	11 00067 00003	Set Index Equal Seven
00460	15 00472 00464	Set u Address of 00464
00461	11 00504 00004	a_8 Into P_i
00462	71 00005 00004	$x \cdot P_i$
00463	54 20000 00046	
00464	35 30474 00004	$P_i + 1$
00465	(23 00464 00073)	
00466	41 00003 00462	
00467	71 00006 00022	$(p - 1) \cdot \ln 2$

$\underline{LOG_e}$

PX 71900-9-(138)

00470	35 00004 20000	ln u^1 Into A Scaled 34
00471	11 00473 00006	
00472	45 00503 00173	Go to Pack
00473	00 00000 00201	
00474	00 00000 00000	a_0 Scaled 34
00475	17 77776 10003	a_1 Scaled 34
00476	70 00101 77550	a_2 Scaled 34
00477	05 23606 17663	a_3 Scaled 34
00500	74 11372 11627	a_4 Scaled 34
00501	02 53533 01102	a_5 Scaled 34
00502	76 36303 74363	a_6 Scaled 34
00503	00 44750 60721	a_7 Scaled 34
00504	77 71310 36772	a_8 Scaled 34

ARCTAN

00505	11 00040 00003	Zero Into 00003
00506	23 00006 00031	Char. of u^1 - 200_8
00507	42 00074 00516	
00510	33 00066 00024	
00511	73 00005 00005	Negative Reciprocal
00512	13 00006 00006	
00513	11 00062 00003	$\pi/2$ Scaled 34 Into 00003
00514	44 00516 00515	
00515	13 00062 00003	$-\pi/2$ Scaled 34 Into 00003
00516	21 00006 00037	
00517	46 00520 00521	

PX 71900-9-(138)

```
00520    11 00040 00005    Replace u[1] by Zero
00521    35 00546 00522
00522    (00 00000 00000)  u[1] or -1/u[1] Scaled 34 Into A
00523    11 20000 00005    x Into 00005 Scaled 34
00524    71 00005 10000
00525    54 20000 00046
00526    11 20000 00006    x^2 Into 00006 Scaled 34
00527    11 00545 00013    Set Index Equal Six
00530    11 00547 00004    C15 Into Pi
00531    15 00544 00534    Set U Address of 00534
00532   [71 00006 00004    x^2 . Pi
00533    54 20000 00046
00534    (35 30556 00004)  Pi + 1
00535    21 00534 00073
00536    41 00013 00532]
00537    71 00005 00004    x . Pi
00540    54 20000 00046
00541    35 00003 00005
00542    54 00005 20001    tan^-1 u[1] Into A Scaled 35
00543    11 00031 00006
00544    45 00550 00173    Go to Pack
00545    00 00000 00006
00546    54 00005 24052
00547    77 73662 40305    C15 Scaled 34
```

PX 71900-9-(138)

00550	00 26305 45073	C_{13} Scaled 34
00551	77 06577 07416	C_{11} Scaled 34
00552	01 42567 67640	C_9 Scaled 34
00553	75 61447 16451	C_7 Scaled 34
00554	03 14201 22666	C_5 Scaled 34
00555	72 52547 44072	C_3 Scaled 34
00556	17 77777 51473	C_1 Scaled 34

00557	11 00042 00676	Set Up Print
		PRINT/PUNCH
00560	44 00561 00562	Print or Punch?
00561	21 00676 00021	Set up Punch
00562	11 00676 00565	Set up Print or Punch at 00565
00563	16 00564 00565	Set up Print or Punch Q
00564	11 00670 10000	"Shift Down" Into Q
00565	(00 00000 00000)	Print/Punch Q
00566	37 00566 (00567)	
00567	11 00010 20000	v^1 Into A
00570	47 00574 00572	If A Zero, Print/Punch Carriage Return
00571	00 00000 00000	Not Used
00572	11 00042 10000	"Carriage Return" Into Q
00573	37 00566 00565	Print/Punch Car. Ret.
00574	11 00016 20000	u^1 Into A
00575	11 00044 10000	"Space" Into Q
00576	46 00577 00600	
00577	11 00061 10000	"_" Into Q

PX 71900-9-(138)

```
00600    37 00566 00565    Print/Punch "Space" or "_"
00601    37 00601 (00602)
00602    47 00605 00603
00603    16 00667 00643    u[1] Equal Zero; Set up Spaces
00604    45 00000 00626
00605    12 00016 00005    Magnitude of u[1] Into 00005
00606    15 00164 00147    Set  up 00005 as U
00607    15 00613 00157    Set up 00671 as V
00610    11 00040 00677    Zero Into 00677
00611    11 00005 20000    N(Packed) Into A
00612    42 00020 00617    If 1> N, Go to 00617
00613    42 00671 00622    If N< 10, Go to 00622
00614    37 00224 00246    Divide by Ten
00615    21 00677 00074    Adjust Decimal Exponent
00616    45 00000 00611
00617    37 00224 00147    Multiply by Ten
00620    23 00677 00074    Adjust Decimal Exponent
00621    45 00000 00611
00622    37 00167 00147    Unpack Normalized N
00623    23 00004 00672
00624    16 20000 00625
00625    54 00003 (30000)  N Into A Scaled 36
00626    11 00040 00005    Set Index to Zero
00627    45 00000 00632
```

```
00630    11 00673 00005    Set Index to Six
00631    71 00003 00674    F · 10₁₀ Into A
00632    11 20000 00003    Fraction Into 00003 Scaled 36
00633    55 00003 00043    Fraction Into 00003 Scaled 35
00634    34 20000 00044
00635    35 00676 00636
00636    (00 00000 00000)  Print/Punch Decimal Digit
00637    41 00005 00631
00640    37 00640 (00641)
00641    11 00675 10000    "." Into Q
00642    37 00566 00565    Print/Punch "."
00643    37 00643 (00644)  Optional Exit for N = 0
00644    37 00640 00630    Translate, Print/Punch Seven More Digits
00645    11 00677 20000    Decimal Exponent Into A
00646    37 00601 00575    Print/Punch "Space" or "_"
00647    12 00677 20000    Magnitude of Exponent Into A
00650    47 00651 00663
00651    73 00060 10000    Translate Exponent
00652    11 20000 00003
00653    11 10000 20000
00654    37 00640 00635    Print/Punch First Digit of Exponent
00655    11 00003 20000
00656    37 00640 00635    Print/Punch Last Digit of Exponent
00657    11 00044 10000    "Space" Into Q
```

PX 71900-9-(138)

00660	37 00566 00565	Print/Punch Space
00661	41 00005 00660	
00662	45 00000 00226	Go to Termination
00663	11 00041 00005	Set Up Print/Punch Three Spaces
00664	45 00000 00657	
00665	11 00060 00005	Set Up Print/Punch Eleven Spaces
00666	45 00000 00657	
00667	00 00000 00665	
00670	00 00000 00057	Shift Down
00671	20 45000 00000	Chip 10.
00672	00 00000 00167	
00673	00 00000 00006	
00674	00 00000 00024	
00675	00 00000 00042	"."
00676	(00 00000 00000)	Print or Punch Temporary Storage
00677	(00 00000 00000)	Decimal Exponent Temporary Storage

CENTRAL EXCHANGE INDEX: _____

TITLE: Expansion of polynomial of form $\sum_{n=0}^{i} a_n x^n$

Programmer: M. A. Thomas

ENTRIES IF MORE THAN ONE OR NOT STANDARD: _____

STANDARD: YES x NO _____

CODING CHECK: x BY PD

MACHINE CHECK: x BY MAT

SELF-RESETTING: YES x NO _____

1. Initial location of argument: Q see miscellany

   Final location of argument: _____

   Location of the function: A

2. ADDRESSES:

   (a) Instructions: 01000 through 01025

   (b) Constants and temporary storage: 01026 through _____

   (c) Constant pool used: 00073, 00074

   (d) Temporary storage pool used 00004-00006

3. INITIAL SETTING OF X:

   (a) Range on x: $x < 1$, $a_i < 1$

   (b) Scaling of x and f(x): $x \cdot 2^{28}$ $f(x) \cdot 2^{28}$

   (c) Brief description of numerical method:

   $P_1 = a_n x + a_{n-1}$

   $P_2 = P_1 x + a_{n-2}$

   $P_i = P_{i-1} x + a_{n-i}$

4. ACCURACY: _____

5. ALARM- CONDITIONS FOR OUT-OF-RANGE TEST:

   Alarm if $\left| \sum a_i x^i \cdot 2^{28} \right| > 2^{35}-1$

PX 71900-9-(139)

6. SPEED:

7. MISCELLANY: Address of a_n in Q_u, address of x in Q_v.
Q_{35} must be 0 or 1 according as descending or ascending order of location of a_i.
Index n (highest power of x) in A_R.
Note: Scale factor may be changed by corresponding change in shift instruction (01011).

01000	37	75700	75702	Alarm
01001	45	00000	(30000)	Exit
01002	16	10000	01024	Entrance
01003	15	10000	01010	Set 01010
01004	16	10000	01010	
01005	36	00074	00004	Form (n-1)
01006	44	01007	01010	
01007	21	01013	01026	
01010	71	(30000)	(30000)	$a_n x$
01011	54	20000	00054	
01012	11	20000	00006	
01013	21	01010	00073	
01014	15	01010	01016	
01015	11	00006	20000	
01016	35	(30000)	00005	$a_n x + a_{n-1}$
01017	43	20000	01021	
01020	45	00000	01000	Overflow alarm
01021	41	00004	01024	Index
01022	11	00005	20000	
01023	45	00000	01001	
01024	71	00005	(30000)	
01025	45	00000	01011	
01026	02	00000	00000	Constant

PX 71900-9-(139

RW-140
FPP-0
Cover Sheet
5/25/55

THE RAMO-WOOLDRIDGE CORPORATION
Los Angeles 45, California

Interpretive Floating Point Package

Identification Tag: FPP-0

This package consists of the following which are written up separately:

1. SNAP 10/10/55
2. SNAP SAMPLER TRACE 3/27/56
3. SNIP (SNAP Complex) 5/1/56

SNAP Sampler Trace

Description

This routine monitors the course of a SNAP program by punching out the results of those SNAP commands which are specified in a list prepared by the programmer. A parameter word will indicate the location of this list.

This list has the following specifications:

a. The list is made up of sublists of four words each. These sublists have the form:

00	FA	LA
00	Np	Ns
00	00000	00000
00	00000	00000

FA is the address where the trace is to start and LA the address at which it is to stop. Np is the number of times FA is to be passed before starting the trace, while Ns is the number of times the section FA to LA is to be traced. The last two words are used by the trace to store the blocked instructions.

b. This list must not be placed in cells 15b thru 107b, but may be put on the drum. In any case, a parameter word 00 Lo Lf will specify its location. Lo is the address of the first word of the list and Lf is the address of the last word. This parameter word must be loaded into cell 71777b. In the event this word is all zero a complete trace of the program is automatically performed.

c. Any number of sublists may be used. A particular address must appear only once in the list since blocking a blocked instruction is not possible.

d. The storage addresses FA and LA must be the addresses in which the instructions to be blocked are actually stored at the time the blocking routine is activated.

The operation of the trace is as follows:

a. When a blocked FA is reached in the program, N_p and N_s are examined. If $N_p = 0$, $N_s \neq 0$ then tracing is initiated and a start indicator placed in the word containing N_p and N_s. On the other hand, if $N_p = 0$ and $N_s \neq 0$ does not occur, then no action is taken.

b. At each SNAP execution, after a trace start, the F register is transferred to the next available position in the trace output hopper (last seven cells in ES). When this hopper contains the results of six SNAP commands (three IP instructions) then the SNAP output routine is used to punch a card containing the information in the trace hopper. The identification field of the output card contains the address of the instruction which produced the numbers in fields one and two. This address is in octal and will be 1725 if the instruction was an FA. The SNAP output command operation is in no way altered during tracing.

c. To empty the trace hopper at any time, start at 72125b. One card will be punched and the machine will stop on MS 0 with PAK = 77777b. SNAP must be in ES to exercise this option.

d. In the event the trace hopper is emptied when it contains no information, a card will be punched containing the address 7777 in the identification field, while the rest of the card will be blank.

e. When a blocked LA is reached in the program and if a start indicator was set up in the word containing the associated N_p and N_s then tracing is stopped and the trace hopper emptied. Otherwise, no action is taken.

f. The execution of the instruction at FA will be traced, but that at LA will not.

g. The seven cells of ES, 1771b - 1777b, cannot be used by the programmer if the trace is to be employed. Normal SNAP operation will not destroy the contents of these cells.

Programming Instructions

1. Load cell 71777b with the parameter word 00 Lo Lf where Lo and Lf are the locations of the first and last words of the list.
2. Load the list in form described above.
3. Start at 72000b. The routine will block the proper instructions as per the list supplied, modify SNAP in order to perform the trace, and then stop with PAK set at 40012b.
4. In the event no parameter word is loaded (and no list is supplied) a start at 72000b will initiate the blocking routine to modify SNAP on MD so that all SNAP commands will be traced. A stop will follow with PAK set at 40012b.

Warnings and Restrictions

1. The list can not occupy cells 15b - 107b.
2. Cells 1771b - 1777b must be reserved for the trace hopper and cannot be used by the programmer.
3. SNAP must be in ES at the time each FA and LA is reached in the program.
4. The instructions in FA and LA must not be read into or out of by the program.
5. Only SNAP commands are traced.
6. The trace can only be used with SNAP and not with the complex version.
7. Activating the trace modifies the copy of SNAP on MD and destroys the complex arithmetic portion of SNAP. To start another program using SNAP or its complex version, it is necessary to restore the library from magnetic tape.

8. An abnormality exists for the following type of a list:

00	FA_1	LA_1
00	N_{p1}	N_{s1}
00	00000	00000
00	00000	00000
00	FA_2	LA_2
00	N_{p2}	N_{s2}
00	00000	00000
00	00000	00000

Assume that N_{p1}, N_{s1}, N_{p2}, N_{s2} are such that tracing in both sublists is concurrent. Further, suppose that FA_1, FA_2, LA_2, LA_1 are executed in the order given. Hence, the trace will be initiated at FA_1 and once again at FA_2. When LA_2 is reached, the trace will stop since it was started at FA_2.

The instructions from LA_2 to LA_1 will not be traced and the trace will be stopped once again when LA_1 is reached. At this time the hopper will be punched. Other unusual combinations can be analyzed in a similar fashion.

THE RAMO-WOOLDRIDGE CORPORATION
Los Angeles 45, California

Interpretive Floating Point Package - Complex

Identification:	SNIP
Type:	Service Routine (with entrance from program available)
Storage:	Cells 634 thru 1023 70000b thru 71662b This includes SNAP The constant pool is used by this routine.
Service Entrance:	Address 40013b
Program Entrance:	See description

Coded by:	C. Koos	
	M. Perry	January, 1956
Code Checked by:	C. Koos	January, 1956
Machine Checked by:	C. Koos	January, 1956
Approved by:	W. F. Bauer	April, 1956

PX 71900-9-(141)

Description

SNIP is the complex arithmetic version of SNAP, a floating point interpretive package. An understanding of the use of SNAP is presupposed.

The activation of this routine changes SNAP into SNIP on the magnetic drum. The original version of SNAP can be obtained again only by a transfer of the Service Routine Library from magnetic tape.

SNIP performs its operations in either real or complex arithmetic depending on a mode which is selected by the programmer, and may be changed at any time.

In the complex mode

1. The complex numbers to be operated on must be in rectangular form, with the real and imaginary parts of the number stored in consecutive cells (For example, the complex number x + iy would be stored in the machine with x in cell α and y in cell $\alpha + 1$).

2. The floating Complex Accumulator requires two cells: Cell 00002, F, is used for the real part; Cell 00003, C, is used for the imaginary part, that is, the two cells 00002 and 00003 constitute the Complex Floating Accumulator, F_c.

3. The Polynomial Multiply command of SNAP is changed so that its execution will result in computing the absolute value of the number stored in F_c.

4. The Fix, Float, Read, Punch and No Operation commands operate exactly as in SNAP, while the remaining commands are changed only in the sense that they now use both cells 00002 and 00003 for the floating accumulator and cells α and $\alpha + 1$ for the argument as explained above.

5. The machine accumulator A contains the real part of the result after the execution of any one of these operations.

6. The Replace and B-box options may be used in all cases that are permitted by SNAP, with two consecutive cells being operated on as described. Keep in mind that the B-box must be indexed by two when used in referencing a list of complex numbers.

7. It is <u>not</u> permissible to load F_c with TP instructions; a Load command must be executed for this purpose.

In the real mode

1. All SNAP commands except Polynomial Multiply operate as in SNAP itself.

2. The execution of the Polynomial Multiply command will give the absolute value of (F_c) just as it does in the complex mode.

SNIP commands

The Complex Accumulator, F_c, is defined as two specific electrostatic cells which contain the complex number x + iy: cell 00002, F, contains x and cell 00003, C, contains y. Both x and y are stored as SNAP numbers; that is, each has its own binary exponent. The notation α_c represents the address of a complex number x + iy, where x is stored at α and y at α + 1, that is, $(\alpha) = x$, $(\alpha + 1) = y$.

The following definitions apply when in the complex mode:

Code	Result
AD 04	$(F_c) + (\alpha_c) \rightarrow F_c$; $(F) \rightarrow A$
SU 10	$(F_c) - (\alpha_c) \rightarrow F_c$; $(F) \rightarrow A$
MP 14	$(F_c) \times (\alpha_c) \rightarrow F_c$; $(F) \rightarrow A$
DV 20	$(F_c) \quad (\alpha_c) \rightarrow F_c$; $(F) \rightarrow A$
PM 24	$\lvert(F_c)\rvert \longrightarrow F_c$; $(F) \rightarrow A$
LD 30	$(\alpha_c) \longrightarrow F_c$; $(F) \rightarrow A$
ST 34	$(F_c) \longrightarrow \alpha_c$; $(F) \rightarrow A$
RT 50	$\sqrt{(F_c)} \longrightarrow F_c$; $(F) \rightarrow A$

The NO (no operation), FI (fix), FL (float), RD (read data) and PD (punch data) instructions of SNAP are unaltered. The PM (polynomial multiply) instruction of SNAP is replaced by the absolute value instruction whether operating in the real or complex arithmetic mode if SNIP has been activated.

Manual Activation of SNIP

1. Insure that both this routine and SNAP are intact on the magnetic drum. (This can be accomplished by a transfer of the Service Routine Library from magnetic tape).

2. Load the problem program - The program should include the Ramo-Wooldridge constant pool, and a jump to start in 40000b, as supplied by RAWOOP.

3. Set PAK to 40013 and start--this changes SNAP into SNIP and causes it to be read into its electrostatic locations, sets the B-box, F, C, and output hopper to zero, supplies an appropriate jump in cells zero and one, positions cards on both sides of the reproducer, and gives control to cell 40000b which normally initiates execution of the problem program. At this time the routine is in the real arithmetic mode.

Programmed Activation of SNIP

Depending on the card positioning desired any one of three different return jump instructions may be used to activate SNIP from the program. Each assumes that there is a manual jump instruction in cell 40000b (such as that supplied by RAWOOP), and in each case control is returned to the instruction immediately following the return jump.

a. Execution of 37 40000 40013b positions cards on both sides of the reproducer.

b. Execution of 37 40000 71644b feeds one card on the punch side of the reproducer.

c. Execution of 37 40000 71646b omits all card positioning.

Otherwise the effect of programmed activation is the same as that described in step 3 under manual activation.

Switching Modes

Activation of SNIP by any one of the methods described above leaves it in the real arithmetic mode. At any time after SNIP has been activated the mode may be switched as follows:

To switch from the real mode to the complex mode execute the return jump

37 01541 01713b

To switch from the complex mode to the real mode execute the return jump

37 01541 01715b

In either case the desired mode change is accomplished, cell 00003, C, is set to zero, and control is returned to the cell immediately following the return jump. The real mode should ordinarily be used wherever possible because it is considerably faster than the complex mode.

Alarms

The SNAP alarm routine is used, with the possibility of the same type of alarm occurring (EW, RT, DV, FI, RD). It is not advisable to continue the problem after an alarm, since either the real or imaginary part of a number may have caused the alarm.

Eigenvector, Eigenvalue Routine

for

Real Symmetric Matrices

Identification Tag:	EGN-0
Type:	Complete package on paper tape and/or binary cards.
Storage:	See storage allocation chart.
Program Entrance:	MD Start
Program Exit:	MS 0 at PAK=VAK=50270b
Alarm Exit:	See section on alarms and stops.
Machine Time:	See section on same.
Mode of Operation:	Fixed point

Coded by:	M. Stein F. Meek	July, 1955, and March, 1956
Machine checked by:	M. Stein F. Meek	August, 1955, and March, 1956
Approved by:	W. Bauer	April, 1956

PX 71900-9-(142)

Description

This package consists of an input routine, a main routine, and an output routine. The input routine reads in the parameters and the elements a_{ij}, $i \geq j$, (the elements on and below the main diagonal) of a real symmetric matrix A. The main routine computes the eigenvalues of A by reducing it to diagonal form with a sequence of orthogonal row and column operations which leave the characteristic equation invariant. The corresponding eigenvectors are computed (also by the main routine) by performing the same sequence of column operations on an identity matrix. The output routine converts and punches on cards the eigenvalues and eigenvectors in fixed point form. The package will handle matrices of order $2 \leq n \leq 38$.

A special mode of operation which requires a minimum number of drum accesses is provided for use with matrices of order $n \leq 23$. The special mode is selected by setting manually selective jump switch 1 to the ON position. Throughout this write-up, T will be defined as the matrix the columns of which are the eigenvectors. In the special mode of operation, the number of rows of T to be obtained can be varied from zero to n and must be specified in advance. Decreasing the number of rows of T to be obtained will allow an increase in the order of the eigenvalue problem which can be solved. Should it be desired to obtain eigenvalues only, the special mode will be able to solve problems of order $n \leq 40$.

With a high speed storage capacity of 4096 words, the ordinary mode of operation will be able to accommodate problems of order $2 \leq n \leq 75$, and the special mode, problems of order $n \leq 50$. If only eigenvalues are to be found, the range of n can be extended to $n \leq 85$. The main routine has been written and the drum storage allocated so as to make the routine easily adaptable to the 1103A.

The main routine does not use the "B register", the "modified multiply add" instruction, or the "polynomial multiply" instruction found only on the Ramo-Wooldridge 1103.

Parameters

1. n is the order of the matrix A.
2. In the special mode of operation, q is equal to n plus the number of rows of T to be obtained. In the ordinary mode of operation, q is automatically set equal to n by the main routine. In the special mode of operation n and q must satisfy the inequality

$$\frac{n^2+n}{2} + (q-n)^2 \leqslant 824.$$

3. N is a positive integer less than 10^5 associated with each matrix A to be run and may be used to identify the output for a given matrix.
4. s is the binary scale factor used to store the matrix elements a_{ij}. s should be chosen such that

$$2^{33} \leqslant n \cdot \max_{i,j} |a_{ij}| \cdot 2^s < 2^{34}$$

See the appendix for more details in regard to scaling.

Preparation of Input Data

The input routine uses CRI-2 to read in the parameters n, q, 10^2N and s, scaled 2^0, and the matrix elements a_{ij}, $i \geqslant j$, scaled 2^s. The first parameter card must contain n with address 00000 and a 12 punch in column 80; for the special mode of operation, it must contain q with address 00001. The second card must contain 10^2N with address 00000, s with address 00001, and a 12 punch in column 80. The third card must contain

a_{11} with address 00000
a_{21} with address 00001
a_{22} with address 00002
a_{31} with address 00003.

The fourth card:

a_{32} with address 00004
a_{33} with address 00005
a_{41} with address 00006
a_{42} with address 00007

and so on. The last card will contain a_{nn} with address $\frac{n(n+1)-2}{2}$ and a 12 punch in column 80. On all cards the unused fields should be left blank. See the CRI-2 write-up (revised 12-9-55) for the details of the card format.

If one wishes to generate the data within the 1103, or if, for any reason, one wishes not to use the input routine, see the appendix.

Switching

Manually selective jump switch number one is used to select either the ordinary mode or the special mode of operation as follows:

MSJ-1 OFF ⟶ ordinary mode.
MSJ-1 ON ⟶ special mode.

Manually selective jump switch number two is used to control a monitoring typeout provided at the end of each sweep[1] as follows:

MSJ-2 OFF ⟶ typeout occurs.
MSJ-2 ON ⟶ typeout suppressed.

Manually selective jump switch number three is used to control the output of the rows of T as follows:

MSJ-3 OFF ⟶ the rows of T are punched.
MSJ-3 ON ⟶ the rows of T are not punched.

The manually selective stop switches are used to provide stops at the end of each of the three segments[2] of the main routine as

(1) A sweep is defined under Mathematical Method.

(2) The main routine is divided into segments I, II, and III. These are described in detail under The Main Routine.

follows:

MSS-1 ON ⟶ stop at conclusion of segment I. A START sets up segment II and jumps to entrance of that segment.

MSS-2 ON ⟶ stop at conclusion of segment II. A START sets up segment I and jumps to entrance of that segment. The setting of MSJ-2 may be changed at this point.

MSS-3 ON ⟶ stop at conclusion of segment III. A START sets up segment I and jumps to entrance of that segment

Operation Instructions

1. The package is available on binary cards and/or on paper tape.
   A. If cards are used, place the package deck, two blank cards, the input cards for the first matrix, two blank cards, the input cards for the second matrix, two blank cards, and so on, ending with two or more blank cards, in the read side of the Bull. Manually "pick a card" and read in the package using CRI-1 (as a service routine).
   B. If paper tape is used, read in the package tape using FRI-0 (as a service routine). Then place the input cards (as in A. above) in the read side of the Bull. Manually "pick a card".
2. Set the MSJ and MSS switches as required. Be sure that there are plenty of blank cards in the punch side of the Bull. Start at 40000b.
3. It will read in the input cards for the first (next) matrix, compute and type out twelve octal digits (with MSJ2 off) after each sweep (see the section on Alarms, Stops for an explanation of the type-out) and finally stop on an MSS-0 at PAK=VAK=50000b. (One may avoid this stop by inserting 45 00000 50000b into 50552b.)
4. Start. It will punch the eigenvalues and, with MSJ-3 off, the rows of T, and stop on an MSS-0 at PAK=VAK=50270b ready to read in the input cards for the next matrix. Set the MSJ and MSS switches for the next matrix and start.
5. Repeat 3 and 4 for the next matrix, and so on.

6. The output cards should be sorted, first on column 80, and then on column 77, and then listed with the SNAP output-fixed point 407 panel.

Alarms, Stops

Since all transformations applied to the matrix A are orthogonal, it can be shown that $S^{(i+1)}$, the sum of the squares of the elements of the lower triangle (including the main diagonal) of data at the end of a sweep ($\frac{n^2-n}{2}$ orthogonal transformations), is equal to $S^{(i)}$, the sum at the beginning of that sweep, plus W, the sum of the squares of all off-diagonal elements deliberately set to zero during the sweep. The double precision quantity $S^{(i+1)} - S^{(i)} - W$ is formed in the accumulator and then scale factored. The scale factor h is tested to see if the number of leading insignificants in $S^{(i+1)} - S^{(i)} - W$ is large enough. If this is not the case the supervisory typewriter will type out an e (for error) followed by the standard typeout which gives the sweep count, the scale factor of $S^{(i+1)} - S^{(i)}$, and the scale factor of $S^{(i+1)} - S^{(i)} - W$. The computer will then halt with PAK=00210b. The sweep may be repeated by depressing the start key. To ignore the alarm set PAK = 00171b and depress the start key.

The scale factor h must satisfy either

$$h \angle (50222b)$$

or

$$h \geqslant 37.$$

It can be proved that the sequence of $S^{(i)}$ is a monotonely increasing sequence which approaches a limit which is less than or equal tc the square of the norm [3] of A. It is believed that this limit is equal to

(3) The norm of A is defined in the appendix under Scaling.

the square of the norm of A, but the equality has never been proved for this process. At the conclusion of each sweep the monotonicity of the sequence is checked by observing the sign of $S^{(i)} - S^{(i+1)}$ If $S^{(i)} - S^{(i+1)} < 0$, the process will continue. Should this not be the case, i.e., if

$$S^{(i)} - S^{(i+1)} \geq 0,$$

the process will be assumed to have converged and the computer will stop on an MSS-0 at PAK=VAK=50000b, ready to enter the output routine.

Arguments unacceptable to the square root subroutine used in the main routine will cause the computer to halt at address 00230b. At this point PAK is set to 00230b. Hence depressing the start key will not cause a resumption of computation.

Since the input routine uses CRI-2, the paragraph under Alarm Conditions of the CRI-2 (revised 12-9-55) write-up applies to the input routine.

If s has been chosen too large (or much too small) a MA overflow may occur or an MSS-0 error stop at PAK=VAK=00062b may occur. See the appendix for more details on this.

Card Output

The output routine uses CPO-2 to convert and punch

1. by rows, the elements d_{ij}, $i \geq j$, of the diagonalized matrix D (the d_{ii} are the eigenvalues of A and the d_{ij}, $i \neq j$ are the off-diagonal elements which have been reduced, essentially, to zero);
2. by rows, the elements of the matrix T.

Each card contains up to six consecutive elements of a row, an identification number, I, and a one-digit card number, C.

The cards containing row i of D will have $I = 10^2N + i$; the cards containing row i of T will have $I = 10^2N + n + i$. All of the cards containing row i of D or T will have the same I and C will run from 1 to k where k is the number of cards required to punch row i, 6 elements per card. The result of all this is that, assuming several matrices have been

run, if the output cards are sorted and listed as in 6, under Operating Instructions, a listing will be obtained of the several matrices in the order of the least significant digit of N. For each matrix, the order of the listing will be: the first six columns of D, the first six columns of T, the second six columns of D, the second six columns of T, and so on. Thus an eigenvector will be listed as a column directly below its corresponding eigenvalue. The 407 panel is so wired that the setting of an alteration switch will cause the paper to eject to start a new sheet at the end of each column.

To bypass the output routine, see the appendix.

Cards, Tapes, Listings, Flow Diagrams

The complete package is available on binary cards (to be read in using CRI-1) and on bi-octal paper tape (to be read in using FRI-0 or the ERA paper tape reader). Also, a flexowriter dump (MDP-0) listing of the entire package is available. RAWOOP-assembled listings are available for the input and output routines only. Detailed flow diagrams for the entire package are on file at the Computing Center.

Mathematical Method

The method used by the main routine is the same method as that described in the write-up of Illinois Code 141-MO in Illiac Library Codes, M-Z. That write-up states "The operations performed in the routine essentially are those described in an unpublished paper by H. H. Goldstine, although there are some modifications".

The method consists of forming the sequence of matrices

$$D_0 = A,$$
$$D_1 = R_1 D_0 R_1',$$
$$\cdots$$
$$D_{i+1} = R_{i+1} D_i R_{i+1}',$$

where R_{i+1} is an orthogonal matrix chosen so as to reduce to zero a pair of off-diagonal elements of D_i. If the sequence of matrices D_i

converge to a diagonal matrix D, i.e., if

$$\lim_{i\to\infty} D_i = D,$$

then the elements of D are the eigenvalues of A, because D_{i+1} is obtained from D_i by a similarity transformation and the characteristic equation remains unchanged. Similarly, if the sequence of matrices T_i, where

$$T_{i+1} = T_i R'_{i+1}, \quad T_0 = I,$$

converges to a matrix T, i.e.,

$$\lim_{i\to\infty} T_i = T,$$

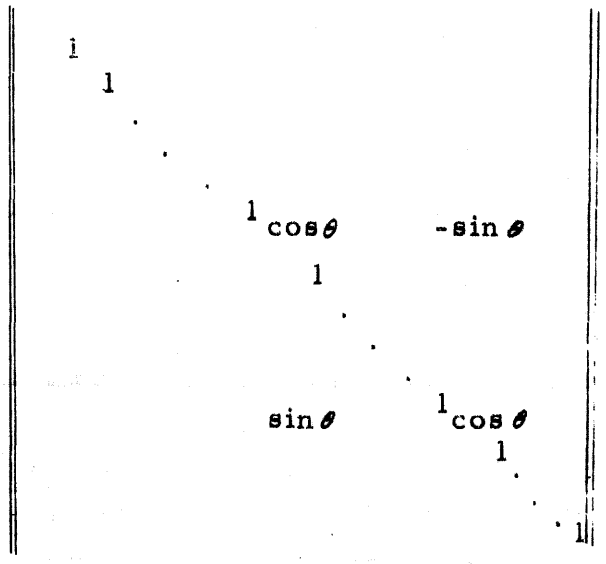
it can easily be shown that the columns of T are the eigenvectors of A, normalized to unit length. The first column of T is the eigenvector corresponding to the eigenvalue in the upper left corner of D, and so on.

If the matrices R_{i+1} are always chosen to reduce the largest (in absolute value) pair of off-diagonal elements of D_i to zero, the method is called the Jacobi[4] method, and it can be proved that the matrices D_i converge, and it can also be proved that if no two eigenvalues of A are equal, then the matrices T_i converge.

The method used by the main routine (and by the Illiac routine) is <u>not</u> to reduce the largest pair of off-diagonal elements to zero at each step, but to reduce the off-diagonal elements to zero in a definite order, namely, working from upper left to lower right below the main diagonal. (One such sequence of reductions is called a sweep.) No proof is known that such a "modified Jacobi method" converges, but it is believed that such a method will converge for all real symmetric matrices.

[4] C. G. J. Jacobi, "Ein leichtes Verfahren, die in der Theorie der Säkularstörungen Vorkommenden Gleichungen numerisch aufzulösen", <u>Jn. reine angew. Math.</u>, V. 30, 1846, p. 51-95.

To reduce the pair of elements $a_{jk} = a_{kj}$ $(k < j)$ to zero the orthogonal transformation R is chosen to be



where the angle θ is such that $\tan 2\theta = a_{jk} / (a_{kk} - a_{jj})$.

Accuracy

Several matrices have been run with the package. Rosser [5] constructed a matrix of order 8 with some of its eigenvalues nearly equal, in order to compare two or three different methods of obtaining eigenvalues and eigenvectors. This matrix was run with the package and the largest eigenvalue error was about $1 \cdot 10^{-5}$, which represents a relative error of $1 \cdot 10^{-8}$. Because of the nearly equal eigenvalues, some of the eigenvectors were less accurate than others. The worst errors were in the components of the vectors corresponding to the three closely-spaced eigenvalues. The largest absolute error was about $3 \cdot 10^{-4}$.

[5] Rosser, Hestenes, Lanczos, and Karush, NBS Journal of Research, v. 47, 1951, pp. 291-296.

Hilbert matrices[6] of orders 8 and 29 were run with the package. The results for the one of order 8 were compared with results obtained on the Illiac (which has 4 more bits than the 1103) and the results for the one of order 29 were compared with results obtained by the Univac at N. Y. U., using Givens' method. For both matrices, all of the eigenvalues agreed through the eighth decimal place (each matrix has for its largest eigenvalue a number slightly less than 2).

Also, several matrices with all elements equal to one were run. Such matrices have n-1 eigenvalues equal to zero and one eigenvalue, λ_1, equal to n, the order of the matrix. For n=40, 8 correct significant digits were obtained for λ_1. For n=38, the same accuracy was obtained for λ_1 and all of the eigenvector components were correct through the 8th decimal place.

Machine Time

The machine times given below are in seconds. They include card input of the data $(2 + \left\{\frac{n^2+n}{8}\right\}$ cards)[7] and card output of the results $(2 + n + (n-6) + (n-12) + \ldots + (n-6k) + (q-n)\left\{\frac{n}{6}\right\}$ cards):[8]

PX 71900-9-(142)

[6] Matrices H with $h_{ij} = \frac{1}{i+j-1}$.

[7] $\{a\}$ means a if a is an integer, b+1 if b is the integral part of a and a is not an integer.

[8] The last term in the series, n-6k, is the last such term which is positive.

n	q	type	mode	number of sweeps	machine time (in seconds)
22	44	all ones	special	2	125
31	62	all ones	ordinary	2	296
38	76	all ones	ordinary	2	471
40	40	all ones	special	2	228
8	16	Rosser	special	6	39
32	64	Rosser	ordinary	5	467
40	40	Rosser	special	6	311
8	16	Hilbert	special	4	29
29	58	Hilbert	ordinary	4	310

The Main Routine

Reading in the package from binary cards or paper tape places the input routine, the three segments of the main routine, and the output routine on the drum.[9] Starting at 40000b (or at 50270b) initiates the execution of the input routine. The input routine places the parameters, n, q, 10^2N, and s, and the elements of the lower triangle of the matrix A on the drum[10] and then initiates the execution of segment III, which runs only once per matrix A. The function of segment III is to set up segments I and II for a specific value of n, make optimizing storage allotments depending on n, and finally to form a suitable matrix of unit vectors from which the eigenvectors can be generated. For the ordinary case (MSJ-1 OFF) the unit vector matrix will be stored in consecutive locations with the first element at address 52000b. In the special case the first element of the unit matrix will be placed immediately following the last element of the lower triangle of A. Segment III concludes itself by block-transferring segment I and data into high speed storage and jumping to the initial line of segment I.

[9] See Storage Allocation for drum storage addresses.

[10] See Storage Allocation for drum storage addresses.

Segment I proceeds through the lower triangle of the matrix being diagonalized taking the elements and the rows in sequence. For each pair of off-diagonal elements an orthogonal matrix R_{i+1} is generated which, when applied along with R'_{i+1} to D_i, produces D_{i+1}, in which the selected pair of elements are zero. In the special mode enough high speed storage (HSS) is available to store both D_i and the matrix T_i of the sequence being transformed into the matrix of eigenvectors. Hence R_{i+1} may be applied directly to T_i at this point and discarded. In the ordinary case HSS capacity is insufficient for the storage of T_i. Hence, enough information must be preserved to reconstruct the sequence of R'_i and apply them to T_i at a later time. This is accomplished by storing pairs $\cos\theta$, $\sin\theta$ sequentially on the drum beginning at address 65000b in a manner corresponding to the order in which the lower triangle was swept through. In order to minimize the number of random drum accesses a HSS region of length $\bar{b} = 01467b - \frac{n^2+n}{2}$ or $\bar{b} = 01470b - \frac{n^2+n}{2}$ (whichever is even) is set aside for temporary storage of the $\cos\theta$, $\sin\theta$ pairs. At the end of one sweep of $(n^2 - n)/2$ transformations R_i during which each pair of off-diagonal elements has been set to zero once, segment I block-transfers segment II into HSS and jumps control to the entrance line of segment II.

In segment II an arithmetic check is made on whether the R_i transformation matrices have been properly generated and applied. If this is the case the latest D_i is dumped on the drum. If a discrepancy is detected the sweep can be repeated. In the ordinary mode of operation segment II will regenerate the R'_i and bring the matrices T_i up to date. Convergence is then checked and if the process has been found to converge, the MSS-0 stop at PAK=VAK=50000b occurs. Otherwise, segment I is set up and the next sweep is initiated.

PX 71900-9-(142)

The Constant Pool

The main routine does not use the R-W constant pool. However, both the input and output routines do use it. It is stored at 50615b and transferred to 00015b just prior to the execution of the input and output

routines.

Storage Allocation (all addresses are octal)

00015 - - 00026	R-W constant pool during execution of the input and output routines.
00027 - - 00040	R-W temporary pool used during execution of the input routine.
00250 - - 00704	Execution addresses and temporary storage for the input routine.
00027 - - 00455 + n 00620 - - 00617 + n	Execution addresses and temporary storage for the output routine.
00000 - - 00307	Execution addresses for segment I and segment II.
00310 - - 00310 + $\frac{n^2+n}{2}$ - 1	Storage for lower triangle of matrix being diagonalized (eigenvalue data) during segment I. Storage for matrix being converted to eigenvectors (eigenvector data) during ordinary mode operation of segment II.
00310 + $\frac{n^2+n}{2}$ - - 01777	Storage for cos θ , sin θ pairs during ordinary mode of operation of both segments I and II. Storage for eigenvector data during special mode of operation.
00401 - - 00551	Execution addresses for segment III.
40000	Storage for a jump to 50270, the entrance of the input routine.
40001 - - 40100	Not used.
40101	Storage for parameter n.
40102	Storage for parameter q.
40103 - - 40407	Storage for segment I.

40410 - - 47777	Dump region for eigenvalue data in ordinary mode. Dump region for both eigenvalue and eigenvector data in special mode. If $n \leq 40$, cells 42100 thru 47777 are not used.
50000	Storage for a jump to 50262, the entrance of the output routine.
50001	Storage for parameter $10^2 N$.
50002	Storage for parameter s.
50003 - - 50257	Storage for segment II.
50260 and 50261	Not used.
50262 and 50263	Storage for the instructions which transfer the output routine into HSS.
50264	Storage for the MSS-0 stop after the output routine, and a jump to 50270, the entrance to the input routine.
50265 - - 50267	Not used.
50270 - - 50273	Storage for the instructions which transfer the input routine and the R-W constant pool into HSS.
50274 - - 50372	Not used.
50373 - - 50551	Storage for segment III.
50552	Storage for the MSS-0 stop after convergence and a jump to 50000.
50553 - - 50614	Not used.
50615 - - 50626	Storage for R-W constant pool.
50627 - - 51014	Storage for CPO-2.
51015 - - 51071	Not used.
51072 - - 51240	Storage for output routine.
51241 - - 51247	Not used.
51250 - - 51256	Storage for input routine.
51257 - - 51262	Not used.
51263 - - 51654	Storage for CRI-2.

PX 71900-9-(142)

51655 - - 51657	Not used.
51660 - - 51750	Storage for the scaling routine, an addition to segment III (see appendix).
51751 - - 51777	Not used.
52000 - - 64777	Not used in special mode. In ordinary mode, storage for eigenvector data. If $n \leq 38$, cells 54644 thru 64777 are not used.
65000 - - 77777	Not used in special mode. In ordinary mode, storage for cos θ, sinθ pairs for one sweep. If $n \leq 38$, cells 67576 thru 77777 are not used.

APPENDIX

Scaling

Under Parameters it was stated that s should be chosen such that

1) $$2^{33} \leq n \max_{i,j} |a_{ij}| \cdot 2^{s} < 2^{34}.$$

Actually, s should be chosen such that

2) $$2^{33} \leq N(A) \cdot 2^{s} < 2^{34},$$

where $N(A) = \sqrt{N^2} = (\sum_{i,j} a_{ij}^2)^{1/2}$ is the norm of A.

If either 1) or 2) is satisfied, one can show that $|a_{kk} - a_{jj}| \cdot 2^{s} < 2^{35}$ (all k, j) for all of the matrices D_i. However, 1) is weaker than 2) in the sense that for many matrices, e.g., Hilbert matrices and Leontieff matrices, 1) will yield an s smaller than 2). With only 35 bits to work with, it may often be important to scale the matrix from 1 to 3 or 4 places further left. Therefore, a scaling routine is included in the package which computes $N^2 \cdot 2^{2s}$ and determines (using the SF instruction) a $\bar{k} \geq 0$ and an $\bar{s} = s + \bar{k}$ such that

3) $$2^{66} \leq N^2 \cdot 2^{2\bar{s}} < 2^{68},$$

which implies that 2) holds for $s = \bar{s}$. Then the a_{ij} which have been scaled 2^{s} (s having been chosen with 1)) are left-shifted $\bar{k}$ places so that the transformations are performed using the "best" scale factor $\bar{s}$. As a matter of fact, all of the matrices mentioned under Accuracy and Machine Time were satisfactorily run using an $\bar{s}$ such that

4) $$2^{68} \leq N^2 \cdot 2^{2\bar{s}} < 2^{70},$$

i.e., an $\bar{s}$ one greater than 2) would indicate should be chosen. This means that, for most matrices, 2) (or 3)) is too strict, and if one is really concerned about obtaining maximum accuracy, one may run a matrix once using 1). Then using the eigenvalues, λ_i, thus obtained,

hand-compute

$$N(A) = \left(\sum_i \lambda_i^2\right)^{1/2}$$

and use,

5) $$2^{34} \leqslant N(A) \cdot 2^{\bar{s}} < 2^{35},$$

which corresponds to 4), to choose $\bar{s} = s$ for a second, more accurate run. On such a second run, the scaling routine should use 4) instead of 3). To accomplish this, insert 00 00000 00043b into 51722b (replacing 00 00000 00041b) after reading in the package. This will avoid the scaling routine error stop (see below).

One more remark should be made in regard to scaling: up to, but not including, one more binary place may be utilized by scaling the a_{ij} by a power of ten, t, chosen so that right-hand inequality 5) is "just barely" satisfied for some $\bar{s}$. (If A has eigenvalues λ_i, $10^t A$ will have eigenvalues $10^t\lambda_i$.)

Scaling Routine Error Stop

If for any reason s has been chosen so large or so small that $N^2 \cdot 2^{2s} \geqslant 2^{68}$ or $N^2 \cdot 2^{2s} < 2^{34}$, an MSS-0 error stop will occur at PAK=VAK=00062b in the scaling routine. In the first case a MA overflow fault may occur during the execution of segment III before entering the scaling routine.

Bypassing the Input Routine

If for any reason one does not wish to use the input routine, one can in some fashion, e.g., using a RAWOOP-assembled data tape, load n, q, 10^2N, and s, all scaled 2^0, into 40101b, 40102b, 50001b, and 50002b, respectively, and a_{11}, a_{21}, a_{22}, a_{31}, a_{32}, a_{33}, ..., a_{n1}, a_{n2}, ..., a_{nn}, all scaled 2^s, into the $(n^2+n)/2$ consecutive cells starting with 40410b. Then instead of starting at 40000b or 50270b, the entrance to the input routine, one should start at 50373b, the entrance to segment III of the main routine. If one wishes to run a series of such matrices, one must

change (50264b) which ordinarily contains an MSS-0 and a jump to 50270b.

Bypassing the Output Routine

If for any reason one does not wish to use the output routine, the following information will be useful: when the process has converged and control has jumped to 50552b, which ordinarily contains the MSS-0 stop with VAK=50000b, the entrance to the output routine, the diagonalized matrix D is stored, scaled $2^{\bar{s}}$, beginning at 40410b. In other words, the original elements $a_{ij} \cdot 2^{s}$ have been replaced by $d_{ij} \cdot 2^{\bar{s}}$ (see Bypassing the Input Routine above). n, q, and $10^{2}N$ are still scaled 2^{0} and are still located at 40101b, 40102b, and 50001b. However, $s \cdot 2^{0}$ has been replaced by $\bar{s} \cdot 2^{30}$ in 50002b. In the ordinary mode, the matrix T will be stored by rows with the first element in 52000b. In the special mode, the first (q-n) rows of T will be stored by rows with the first element in $(40410 + \frac{n^2+n}{2})$ b. In either mode, all elements of T will be scaled 2^{34}, and the columns of T, the eigenvectors, will be normalized to unit length.

PX 71900-9-(142)

THE RAMO-WOOLDRIDGE CORPORATION
Los Angeles 45, California

Floating Point Gill Method

Identification Tag:	NUI-4
Type:	Subroutine
Assembly Routine Spec:	SUB 51921 08914
Storage:	89 words total program storage 9 words temporary pool used, addresses 27 b thru 37 b The constant pool is used.
Entrance and Exit:	RJ GIL01 GIL02 set up RJ GIL01 GIL03 to get next point MJ 0 GIL04 From derivative calculation
Machine Time:	Approximately (9.7 + 84.6n) m.s. per time interval, where n equals the number of equations in the system.

Coded by:	J. Carlson	May, 1956
Approved by:	W. F. Bauer	May 10, 1956

PX 71900-9-143

Description

The Gill Method Subroutine integrates a system of first order, differential equations using a step-by-step process. Using the values of the variables at a point and the coding for computing the derivative of each of the dependent variables at that point, the Gill Method Subroutine produces the coordinates for the next point of the solution each time it is entered.

A special entrance sets up the subroutine for a particular system of equations, thus allowing the subroutine to solve concurrently several different systems in the same program.

The independent variable is incremented within the subroutine itself.

Notation

The system of equations to be solved is

$$\frac{dy_i}{dx} = f_i\,(x, y_1, y_2, \ldots, y_n),\ (i = 1, 2, \ldots, n).$$

n is the number of equations in the system.

q_i are intermediate values of the calculation (zero initially)

x is the increment of the independent variable x

Programming and Operating Instructions

SNAP must be in E.S.

Assign the Gill Method Subroutine to some arbitrary region, say GILOO. This region need not be located in the low numbered half of E.S.

In order to solve a given system, the following array of variables, derivatives, intermediate values, and parameters should be assigned a region, say DEQOO. Although the programmer will undoubtedly desire to have this region located in the low numbered half of E.S., it is not necessary for the operation of this subroutine.

DEQOO	n	Fixed point form scaled 2^0.
DEQOl	Δx	Floating point form
DEQO2	x	"
DEQO3	$\frac{dy_1}{dx}$	"
DEQO4	y_1	"
DEQO5	q_1	"

PX 71900-9-(143)

DEQ06 $\frac{dy_2}{dx}$

DEQ07 y_2

DEQ08 q_2

In addition, the coding for computing $\frac{dy_i}{dx}$ for all i, (i = 1, 2, . . ., n) should be assigned a region, say DFQ00. This coding will use the values in region DEQ00 to compute all $\frac{dy_i}{dx}$ as specified by the equations in the system and should place the results in the appropriate places in region DEQ00. It should then exit to the Gill Method Subroutine with an MJ 00000 GIL04 (see below).

Assuming the Gill Method Subroutine is in region GIL00, the three entrances are GIL02, GIL03, and GIL04. The exit is GIL01.

The first entrance, GIL02, is used for setting up the Gill Method Subroutine only for the particular system to be solved. It is entered by an RJ command followed by a parameter word which specifies the location of the variables, and the location of the coding for calculating the derivatives:

```
RJ  GIL01  GIL02
00  DEQ00  DFQ00
```

The second entrance, GIL03, is the entrance for producing a point of the solution. It is entered by an RJ command: RJ GIL01 GIL03. Entering using this command results in four passes through both the Gill Method Subroutine and the coding for computing the derivatives, and leaves in region DEQ00 the new values of the variables, the derivatives at those values, and x advanced by Δx, ready for the next step.

The third entrance, GIL04, is the entrance from the coding for calculating the derivatives and is used on each of the four passes necessary for computing one point. As noted above, it is entered by an MJ command in the DFQ00 region:

```
MJ  00000  GIL04
```

Mathematical Analysis

Theory. "A Process for the Step-by-Step Integration of Differential Equations in an Automatic Digital Computing Machine" by S. Gill, published in Cambridge Philosophical Society Proceedings, Vol. 47, Part I, January 1951, should be consulted for a detailed analysis of the process on which the subroutine is based.

Suppose we know the point $(X, Y_1, Y_2, \ldots, Y_n)$ on the curve defined by the system of equations

$$\frac{dy_1}{dx} = f_1(x, y_1, y_2, \ldots, y_n)$$

$$\frac{dy_2}{dx} = f_2\,(x;\ y_1,\ y_2,\ \ldots,\ y_n)$$

$$\vdots$$

$$\frac{dy_n}{dx} = f_n\,(x,\ y_1,\ y_2,\ \ldots,\ y_n)$$

The Gill Method is a process by which we can find the next point on the curve: i.e. the value of $y_1, y_2, \ldots, y_n$ for $x = X + h$.

The process can be better understood if the case where $n = 1$ is first considered.

We have the point (X,Y) on the curve $\frac{dy}{dx} = f(x,y)$, and we want to find y at $X + h$; i.e. we want $k = \delta y$ such that $\left[\frac{dy}{dx}\right]_{X+h,\ Y+k} = f(X + h,\ Y + k)$.

We derive k by making four approximations and averaging them in a particular way.

First approximate the curve by a straight line through (X,Y) with the slope $\left[\frac{dy}{dx}\right]_{X,Y} = f(X,Y)$, and find a first approximation to k:

$$k_o = h \cdot f(X,Y)$$

Then we travel a fraction m of the way along this line to the point $(X+mh,\ Y+mk_o)$ and find $f(X + mh,\ Y + mk_o)$.

This gives us a new straight line through $(X + mh,\ Y + mk_o)$ with slope $f(X + mh,\ Y + mk_o)$, and we find

$$k_1 = h\ f(X + mh,\ Y + mk_o)$$

We now use k_o and k_1 to find a third point at which f is calculated: $(X + nh,\ Y + [n-r]\ k_o + rk_1)$.

$$k_2 = h\ f(X + nh,\ Y + [n-r]\ k_o + rk_1)$$

Similarly,

$$k_3 = h \cdot f(X + ph,\ Y + [p-s-t]\ k_o + sk_1 + tk_2$$

The weighted average of k_o, k_1, k_2, and k_3 is the desired $k = \delta y$:

$$\delta y = y(X + h) - y(X) = c_o k_o + c_1 k_1 + c_2 k_2 + c_3 k_3$$

where $c_o + c_1 + c_2 + c_3 = 1$.

For a system of equations, the same four steps given above are made for each equation and

$$\delta y_i = c_o k_{io} + c_1 k_{i1} + c_2 k_{i2} + c_3 k_{i3} \text{ where } c_o + c_1 + c_2 + c_3 = 1.$$

The above process is, for certain values of m, n, p, s, t, c_o, c_1, c_2, and c_3, the Runge-Kutta process. The Gill process was derived, with application to machine use in mind, by minimizing the number of storage cells required. For the Gill Method the above constants are

$$m = 1/2, \qquad r = 1 - \sqrt{1/2}, \qquad c_o = 1/6$$

$$n = 1/2, \qquad s = -\sqrt{1/2}, \qquad c_1 = (1/3)(1 - \sqrt{1/2})$$

$$p = 1, \qquad t = 1 + \sqrt{1/2}, \qquad c_2 = (1/3)(1 + \sqrt{1/2})$$

$$c_3 = 1/6$$

The Gill process further systematizes the calculation so as to increase the accuracy and simplify the coding.

The Subroutine. As used in the Gill Method Subroutine, the process is as follows:

It is assumed that the $f_i(x, y_{10}, y_{20}, \ldots, y_{no})$ and the y_i are available.

1st pass:

Advance x by (1/2)h

$k_{io} = h \cdot f_i (x, y_{10}, y_{20}, \ldots, y_{no})$

$r_{il} = (1/2)k_{io} - q_{io}$

$q_{il} = q_{io} + 3r_{il} - (1/2)k_{io}$

$y_{il} = y_{io} + r_{il}$

Calculate $f_i (x, y_{11}, y_{21}, \ldots, y_{nl})$ in programmer's own coding.

2nd pass:

$k_{il} = h\, f_i (x, y_{11}, y_{21}, \ldots, y_{nl})$

$r_{i2} = (1 - \sqrt{1/2})(k_{il} - q_{il})$

$q_{i2} = q_{il} + 3r_{i2} - (1 - \sqrt{1/2})k_i$

$y_{i2} = y_{il} + r_{i2}$

Calculate $f_i (x, y_{12}, y_{22}, \ldots, y_{n2})$ in programmer's own coding.

3rd pass:

Advance x by (1/2)h

$$k_{i2} = h \cdot f_i\ (x, y_{12}, y_{22}, \ldots, y_{n2})$$

$$r_{i3} = (1 + \sqrt{1/2})\ (k_{i2} - q_{i2})$$

$$q_{i3} = q_{i2} + 3r_{i3} - (1 + \sqrt{1/2})\ k_{i2}$$

$$y_{i3} = y_{i2} + r_{i3}$$

Calculate $f_i\ (x, y_{13}, y_{23}, \ldots y_{n3})$

4th pass:

$$k_{i3} = h \cdot f_i\ (x, y_{13}, y_{23}, \ldots y_{n3})$$

$$r_{i4} = (1/6)(k_{i3} - 2q_{i3})$$

$$q_{i4} = q_{i3} - 3r_{i4} - (1/2)k_{i3}$$

$$y_{i4} = y_{i3} + r_{i4}$$

Calculate $f_i\ (x, y_{14}, y_{24}, \ldots, y_{n4})$ in programmer's own coding.

Errors. The paper by S. Gill mentioned previously includes a detailed analysis of errors, both truncation error and round-off error.

The expression for the truncation error in δy_i is too complicated to give here, but its dominating term, the author states, is

$$\frac{h^5}{-120} \sum_{\substack{o \\ j,k,l,m}}^{n} \left[f_j \frac{\partial f_k}{\partial y_j} \cdot \frac{\partial f_l}{\partial y_k} \cdot \frac{\partial f_m}{\partial y_l} \cdot \frac{\partial f_i}{\partial y_m} \right]_{x=X} \qquad \text{where } y_o = x,\ f_o = 1,$$

and the truncation error in δy_i will be approximately this when the second partial derivatives are all close to zero. It is probably more useful to say merely that the truncation error is of the order of h^5.

The standard deviation in $y_i - (1/3)q_i$ over one step from all rounding off errors is (where f is the quantity mentioned in the section on notation)

$$1/6 \left[7/3 \left\{ 2^{-2f} + (1/16)h^2 \sum_j \left(\frac{\partial f_i}{\partial y_j} \right)_X^2 \right\} \right]^{1/2}$$

u, u = the value of one unit in the last digit of y.

PX 71900-9-(143)

Machine Checking

The following system of two equations was solved using this routine:

$$\frac{dy_1}{dx} = \cos x$$

$$\frac{dy_2}{dx} = -\sin x$$

The initial conditions, at $x = 0$, were

$$y_1 = 0 \quad \text{and} \quad y_2 = 1$$

The interval, Δx, used was $2\pi/360$ radians. At $x = 360^\circ$ the results were accurate to 8 decimal digits.

```
D           GIL00 01024
D           GIM00 51921
GIM00    MS 00000 GIL00
GIM01    MJ 00000 00000
GIM02    MJ 00000 GIL49
GIM03    MJ 00000 GIL08
GIM04    RA GIL71 00016
GIM05    EJ GIL69 GIL73
GIM06    EJ GIL68 GIL01
GIM07    MJ 00000 GIL18
GIM08    TP 00016 GIL71
GIM09    TU GIL72 GIL20
GIM10    TP 00000 GIL88
GIM11    TP 00000 00024
GIM12    TP GIL88 00002
GIM13  ADNO 00024 00000
GIM14    TP 00002 GIL87
GIM15    TP GIL75 00031
GIM16  ADMP 00024 00031
GIM17    TP 00002 00000
GIM18    TV 00000 GIL70
GIM19    RP 30003 GIL21
GIM20    TP 00000 00023
GIM21    RA GIL20 GIL67
GIM22    TU GIL66 GIL25
GIM23    TV GIL66 GIL44
GIM24    RP 30003 GIL26
GIM25    TP 00000 00026
GIM26    TP 00024 00002
GIM27  MPNO 00028 00000
GIM28    TP 00002 00029
GIM29    TP GIL88 00002
GIM30  MPNO 00026 00000
GIM31    TP 00002 00030
GIM32  MPSU 00023 00029
GIM33    TP 00002 00029
GIM34  ADNO 00027 00000
GIM35    TP 00002 00027
GIM36    TN 00025 00002
GIM37  MPAD 00030 00028
GIM38    TP 00002 00030
GIM39    TP GIL85 00031
GIM40    TP 00029 00002
GIM41  DVAD 00031 00030
GIM42    TP 00002 00028
```

```
02000  00 00000 00000
67221  00 00000 00000
67221  56 00000 02000
67222  45 00000 00000
67223  45 00000 02061
67224  45 00000 02010
67225  21 02107 00020
67226  43 02105 02111
67227  43 02104 02001
67230  45 00000 02022
67231  11 00020 02107
67232  15 02110 02024
67233  11 00000 02130
67234  11 00000 00030
67235  11 02130 00002
67236  14 04030 00000
67237  11 00002 02127
67240  11 02113 00037
67241  14 04030 14037
67242  11 00002 00000
67243  16 00000 02106
67244  75 30003 02025
67245  11 00000 00027
67246  21 02024 02103
67247  15 02102 02031
67250  16 02102 02054
67251  75 30003 02032
67252  11 00000 00032
67253  11 00030 00002
67254  14 14034 00000
67255  11 00002 00035
67256  11 02130 00002
67257  14 14032 00000
67260  11 00002 00036
67261  14 14027 10035
67262  11 00002 00035
67263  14 04033 00000
67264  11 00002 00033
67265  13 00031 00002
67266  14 14036 04034
67267  11 00002 00036
67270  11 02125 00037
67271  11 00035 00002
67272  14 20037 04036
67273  11 00002 00034
```

PX 71900-9-(143)

```
GIM43   RP 30003 GIL45
GIM44   TP 00026 00000
GIM45   RA GIL25 GIL67
GIM46   RA GIL44 GIL69
GIM47   RS GIL70 00016
GIM48   ZJ GIL24 00000
GIM49   TP GIL01 A0000
GIM50   LA A0000 00015
GIM51   TU A0000 GIL52
GIM52   TP 00000 A0000
GIM53   TV A0000 GIL48
GIM54   TU A0000 GIL18
GIM55   AT 00015 A0000
GIM56   TU A0000 GIL10
GIM57   AT 00015 A0000
GIM58   TU A0000 GIL11
GIM59   TU A0000 GIL66
GIM60   LA A0000 00057
GIM61   TV A0000 GIL17
GIM62   TV A0000 GIL66
GIM63   RA GIL66 00017
GIM64   RA GIL01 00016
GIM65   MJ 00000 GIL01
GIM66   00 00000 00000 B
GIM67   00 00003 00000 B
GIM68   00 00000 00005 B
GIM69   00 00000 00003 B
GIM70   00 00000 00000 B
GIM71   00 00000 00000 B
GIM72   00 GIL75 00000 BRB
GIM73   TP GIL87 00002
GIM74   MJ 00000 GIL17
GIM75   05 00000 00000 -01 F
GIM76   01 00000 00000  00 F
GIM77   05 00000 00000 -01 F
GIM78   02 92893 21881 -01 F
GIM79   02 92893 21881 -01 F
GIM80   02 92893 21881 -01 F
GIM81   01 70710 67812  00 F
GIM82   01 70710 67812  00 F
GIM83   01 70710 67812  00 F
GIM84   01 66666 66667 -01 F
GIM85   03 33333 33333 -01 F
GIM86   05 00000 00000 -01 F
GIM87   00 00000 00000 B
GIM88   00 00000 00000 B
START
```

```
67274   75 30003 02055
67275   11 00032 00000
67276   21 02031 02103
67277   21 02054 02105
67300   23 02106 00020
67301   47 02030 00000
67302   11 02001 20000
67303   54 20000 00017
67304   15 20000 02064
67305   11 00000 20000
67306   16 20000 02060
67307   15 20000 02022
67310   35 00017 20000
67311   15 20000 02012
67312   35 00017 20000
67313   15 20000 02013
67314   15 20000 02102
67315   54 20000 00071
67316   16 20000 02021
67317   16 20000 02102
67320   21 02102 00021
67321   21 02001 00020
67322   45 00000 02001
67323   00 00000 00000
67324   00 00003 00000
67325   00 00000 00005
67326   00 00000 00003
67327   00 00000 00000
67330   00 00000 00000
67331   00 02113 00000
67332   11 02127 00002
67333   45 00000 02021
67334   20 04000 00000
67335   20 14000 00000
67336   20 04000 00000
67337   17 74537 30314
67340   17 74537 30314
67341   17 74537 30314
67342   20 16650 11714
67343   20 16650 11714
67344   20 16650 11714
67345   17 65252 52525
67346   17 75252 52525
67347   20 04000 00000
67350   00 00000 00000
67351   00 00000 00000
        45 00000 00000
```

PX 71900-9-(143)

THE RAMO-WOOLDRIDGE CORPORATION
Los Angeles 45, California

FLOATING POINT SINE-COSINE

Specifications

Identification Tag:	SIN-4
Type:	Subroutine
Assembly Routine Spec:	SUB 51856 06510
Storage:	65 words total program storage
	4 words temporary storage pool used, addresses 27 b through 32 b.
	The constant pool is used by this routine.
Entrance and Exit:	RJ SUB01 SUB02 for the sine
	RJ SUB01 SUB03 for the cosine
Machine Time:	3.9 ms average, 4.8 ms maximum

Coded by:	M. Perry	May, 1956
Approved by:	W. Bauer	May 15, 1956

PX 71900-9-(144)

Description

When supplied with an argument X in SNAP form, this routine will evaluate sin X or cos X (depending on which of the two entrances is used) using a Rand Polynomial Approximation, producing the answer in SNAP form.

Programming Instructions

This routine can be inserted into a program by CMP-0 by the use of a "SUB" card in the input deck.

1. Place the double length extension of X in the accumulator.

   X must be in radians and must be in SNAP form.

2. Return jump to the subroutine. Assuming that the subroutine was assigned to region SUB00 for assembly, use either the instruction RJ SUB01 SUB02 for the sine, or the instruction RJ SUB01 SUB03 for the cosine.

3. At the time of exit from the subroutine, the double length extension of sin X (or cos X) in SNAP form will be in the accumulator.

Error Analysis

Sin X or cos X is computed to 26 bits of accuracy or to as many correct bits as there are in the Fractional portion of X, whichever is less. For $X \geq 2^{27}$, this routine substitutes zero for the argument. The alarm exit is not used.

Mathematical Method

1. Let $y = (2/\pi)X$, then $\sin X = \sin(\pi/2)(y)$

   $$\cos X = \sin(\pi/2)(y + 1)$$

2. Divide y (or y + 1) into an integral part R, and a fractional part S.

3. R defines the quadrant into which X falls. Let R' be the two low order positions of R, since in binary notation, any other positions merely define a number of complete revolutions.

4. R' is a number one less than the number of the quadrant into which X falls.

5. S defines the displacement (in a position direction) within the quadrant indicated by R'.

6. Therefore, if R' = 00 Let Z = S first quadrant
   R' = 01 Let Z = (1-S) second quadrant
   R' = 10 Let Z = (-S) third quadrant
   R' = 11 Let Z = (1-S) fourth quadrant

PX 71900-9-(144)

7. Sin (or cos) X = $\sin(\pi/2)Z$.

8. $(1/z)\sin(\pi/2)z$ is approximated by the Rand Polynomial Approximation Number 16, using argument z.

9. If $x < 1/2$, $(2/\pi)x$, which is in floating form, is substituted for z before doing step 10.

10. Multiply the approximation from step 8 by z giving the result, sin x (or cos x).

Range of Variable

No alarm condition is recognized by this routine. However, as X approaches $\pm 2^{27}$ the number of significant digits in Sine X (or Cosine X) approaches zero and X merely defines a number of revolutions and does not significantly designate an angle.

```
D          02S00 00023                                  00027  00 00000 00000
D          00S00 01024                                  02000  00 00000 00000
D          01S00 01079                                  02067  00 00000 00000
D          D0S00 51856                                  67120  00 00000 00000
D          D1S00 51911                                  67207  00 00000 00000
D0S00   RJ 00000 00000  ALARM                           67120  37 00000 00000
D0S01   MJ 00000 00000  NORMAL EXIT                     67121  45 00000 00000
D0S02   RP 20002 00S04  SIN ENTRY                       67122  75 20002 02004
D0S03   TP 00013 02S04  COS ENTRY                       67123  11 00015 00033
D0S04   TU 00S02 00S51  SET FOR POS                     67124  15 02002 02063
D0S05   LA A0000 00008                                  67125  54 20000 00010
D0S06   TM B0000 02S00  EXP PLUS 200                    67126  12 30000 00027
D0S07   LA A0000 00001                                  67127  54 20000 00001
D0S08   LQ A0000 00035                                  67130  55 20000 00043
D0S09   MP Q0000 01S05                        69        67131  71 10000 02074
D0S10   TP B0000 Q0000                        34        67132  11 30000 10000
D0S11   TP Q0000 02S01                                  67133  11 10000 00030
D0S12   RS 02S00 01S08  EXP                             67134  23 00027 02077
D0S13   SJ 00S14 00S21                                  67135  46 02016 02025
D0S14   SA 01S07 00000                                  67136  32 02076 00000
D0S15   SJ 00S18 00S16                                  67137  46 02022 02020
D0S16   AT 00S53 00S17                                  67140  35 02065 02021
D0S17   LA Q0000 00007                                  67141  54 10000 00007
D0S18   TP B0000 Q0000                                  67142  11 30000 10000
D0S19   TP 02S04 A0000                                  67143  11 00033 20000
D0S20   ZJ 00S36 00S26                                  67144  47 02044 02032
D0S21   TJ 01S07 00S23                                  67145  42 02076 02027
D0S22   CC Q0000 A0000                                  67146  27 10000 20000
D0S23   TV A0000 00S24                                  67147  16 20000 02030
D0S24   LA Q0000 00000                                  67150  54 10000 00000
D0S25   IJ 02S04 00S28  SIN                             67151  41 00033 02034
D0S26   TN Q0000 A0000  COS                             67152  13 10000 20000
D0S27   AT 01S06 Q0000                                  67153  35 02075 10000
D0S28   QT 01S06 02S01                                  67154  51 02075 00030
D0S29   CC 02S00 A0000                                  67155  27 00027 20000
D0S30   QJ 00S31 00S32                                  67156  44 02037 02040
D0S31   RS 00S51 00015                                  67157  23 02063 00017
D0S32   QJ 00S33 00S35                                  67160  44 02041 02043
D0S33   TP 01S06 A0000                                  67161  11 02075 20000
D0S34   ST 02S01 02S01                                  67162  36 00030 00030
D0S35   TP 02S01 Q0000                                  67163  11 00030 10000
```

```
D0S36   MP  Q0000  Q0000                                  67164  71 10000 10000
D0S37   SA  01S06  00001                                  67165  32 02075 00001
D0S38   TP  B0000  02S02          SQUARED    34            67166  11 30000 00031
D0S39   PM  01S01  01S00                                  67167  24 02070 02067
D0S40   RP  20003  00S42                                  67170  75 20003 02052
D0S41   PM  01S02  02S02                      69          67171  24 02071 00031
D0S42   MP  B0000  02S01                      68          67172  71 30000 00030
D0S43   TP  B0000  A0000          FINAL MANT33            67173  11 30000 20000
D0S44   ZJ  00S45  00S01                                  67174  47 02055 02001
D0S45   SF  A0000  00S54                                  67175  74 20000 02066
D0S46   LA  A0000  00027                                  67176  54 20000 00033
D0S47   TP  B0000  Q0000                                  67177  11 30000 10000
D0S48   RA  02S00  00S54                                  67200  21 00027 02066
D0S49   SA  01S09  00027                                  67201  32 02100 00033
D0S50   CC  A0000  Q0000                                  67202  27 20000 10000
D0S51   RP  00000  00S01                                  67203  75 00000 02001
D0S52   TN  A0000  A0000                                  67204  13 20000 20000
D0S53   LA  Q0000  00007                                  67205  54 10000 00007
D0S54   00  00000  00000                                  67206  00 00000 00000
D1S00   01  51484  19000  -04  38  C9                     67207  00 02366 57351
D1S01   -4  67376  55700  -03  36  C7                     67210  77 54666 31633
D1S02   07  96896  79280  -02  35  C5                     67211  02 43150 53663
D1S03   -6  45963  71106  -01  34  C3                     67212  65 52420 76451
D1S04   01  57079  63185        33  C1                    67213  14 44176 65102
D1S05   06  36619  77225  -01  34  2 OVER PI              67214  12 13714 06667
D1S06   17  77777  77777  B        MASK                   67215  17 77777 77777
D1S07   00  00000  00034  B        28                     67216  00 00000 00034
D1S08   00  00000  00200  B        128                    67217  00 00000 00200
D1S09   00  00000  00072  B                               67220  00 00000 00072
START                                                            45 00000 00000
```

PX 71900-9-(144)

THE RAMO-WOOLDRIDGE CORPORATION

LOS ANGELES 45, CALIFORNIA

Standard Atmosphere Calculation

Specifications

Identification Tag:	ATM-1
Type:	Subroutine
Assembly Routine Spec:	SUB 50776 13253
Storage:	79 instructions, addresses 1KL00 thru 1KL11 1KR00 thru 1KR33 1KP00 thru 1KP32 53 constants in program, addresses 1KE00 thru 1KE36 1KF00 thru 1KF15 132 words total program storage. 7 words temporary storage pool used, addresses OKT00 thru OKT06 The constant pool is used by this routine.
Program Entrance:	(See write-up.)
Program Exit:	Address OKL01
Alarm Exit:	The alarm exit is used by this routine. OKL00
Drum Assignment:	Address 65030b thru 65233b
Machine Time	1.34 ms FOR TEMPERATURE 2.52 ms FOR SPEED OF SOUND 4.40 ms FOR DENSITY 5.48 ms FOR PRESSURE 6.76 ms FOR ALL 4 QUANTITIES
Mode of Operation:	Fixed Point
Coded by:	M. Elmore — August, 1955
Code Checked by:	L. Kimble — April, 1956
Machine Checked by:	L. Kimble — April, 1956
Approved by:	W. F. Bauer — April, 1956

Description

Given the geometric altitude Y in feet ($0 \leqslant Y < 1,048,576$), this routine will compute the pressure, P, in pounds per square inch, density ρ, in slugs per cubic foot, speed of sound, V, in feet per second, or temperature, T, in degrees Rankine, or all four quantities at that altitude, depending on which one of five possible entrances is used.

Each of these quantities is calculated from a formula derived by fitting curves to data from the ICAO Interim Standard Atmosphere Table.

Programming Instructions

1. Place $Y \cdot 2^{15}$ in the accumulator.

2. RJ to the subroutine.

Assume that the subroutine has been assigned some arbitrary region, 0KL00. Then the exit from the subroutine will be at 0KL01. Depending on the entrance used, the results are left in the accumulator or stored in cells 24, 25, 26, or 27 of the temporary storage pool, as follows:

Enter With:	$P \cdot 2^{30}$ lbs. per $in.^2$	$\rho \cdot 2^{39}$ slugs per $ft.^3$	$V \cdot 2^{21}$ ft. per sec.	$T \cdot 2^{21}$ degrees Rankine	Machine Time ms
RJ 0KL01 0KL02	Acc.	25	---	27	5.48
RJ 0KL01 0KL03	---	Acc.	---	---	4.40
RJ 0KL01 0KL04	---	---	Acc.	27	2.52
RJ 0KL01 0KL05	---	---	---	Acc.	1.34
RJ 0KL01 0KL06	24	25	26	27	6.76

Mathematical Analysis

Temperature, T.

The temperature as given by the ICAO Table is essentially a linear function of the altitude in six distinct regions. In region 1, from 0 to 11 km, temperature steadily decreases; in region 2, from 11 to about 25 km, it is constant; in region 3, from about 25 to 48 km, it increases steadily; in region 4, from about 48 to about 53 km, it is constant again; in region 5, from about 53 km to about 77 km, it again decreases; and in region 6, from about 77 km to the end of the range, it is constant. In each of these regions, the routine calculates

$T = B_o + B_1 Y$ where Y is the geometric altitude, and B_o and B_1 are constants, different for each of the six regions.

Speed of sound, V

The speed of sound is directly proportional to the square root of the temperature. In order to calculate the speed of sound, the routine calculates the temperature and then obtains a close approximation to its square root.

Density, ρ

The density is approximated by a function of the form 2 raised to the power $(A_0 + A_1 H + A_2 H^2)$ where H is the geopotential altitude and A_0, A_1, A_2 are constants, different for each of the six regions described under temperature.

Pressure, P

The pressure is proportional to the product of the density times the temperature; in order to find the pressure, the routine first finds the density and temperature

For altitudes greater than 83 km, the pressure and density are set equal to zero, and the speed of sound and temperature are set to constant values of 886.7 and 363.4 respectively.

Table of Constants:

Region	A_0	A_1	A_2	B_0	B_1
1	-8.718341	$-.4152989 \times 10^{-4}$	$-.1910522 \times 10^{-9}$	518.65	$-.3560071 \times 10^{-2}$
2	-7.965339	$-.6934172 \times 10^{-4}$	ZERO	390.2	ZERO
3	-6.745764	$-.941444 \times 10^{-4}$	$.1203316 \times 10^{-9}$	256.039	$.162733 \times 10^{-2}$
4	-10.210967	$-.5315949 \times 10^{-4}$	ZERO	508.79	ZERO
5	-15.1932	$-.2626978 \times 10^{-5}$	$-.1260831 \times 10^{-9}$	838.663	$-.188169 \times 10^{-2}$
6	-4.806382	$-.7562816 \times 10^{-4}$	ZERO	363.4	ZERO

PX 71900-9-(145)

Alarm Conditions

If the subroutine is entered with a negative argument ($Y < 0$) it will enter the alarm routine ALR-1: "ATM-1" and the octal address of the RJ order with which ATM-1 was entered will print out on the flexowriter.

Machine Checking

A driver routine was written to obtain values of pressure, density, speed of sound and temperature at 0, 1, 2, ... 91 km. in (geopotential) altitude and these were checked against values based on the ICAO Interim Standard Atmosphere Table.

The greatest relative error in pressure for the lower altitudes is .002. While the relative error for altitudes in regions 4, 5, and 6 increases sharply, the absolute error decreases and is less than .00008. The greatest absolute error in density is very small for the entire range and is of the order of magnitude of .0000 0004. The greatest relative error in the speed of sound for the entire range is .0002. The greatest relative error in temperature is .0001.

D		0KL00	01024			02000	00	00000	00000
D		1KL00	50776			65030	00	00000	00000
D		0KR00	01036			02014	00	00000	00000
D		1KR00	50788			65044	00	00000	00000
D		0KP00	01070			02056	00	00000	00000
D		1KP00	50822			65106	00	00000	00000
D		1KE00	50855			65147	00	00000	00000
D		1KF00	50892			55214	00	00000	00000
D		0KE00	01103			02117	00	00000	00000
D		0KF00	01140			02164	00	00000	00000
D		0KT00	00023			00027	00	00000	00000
1KL00	37	75701	75702	B	ALARM EXIT	65030	37	75701	75702
1KL01	MJ				EXIT	65031	45	00000	00000
1KL02	MJ		0KP23		PRESSURE	65032	45	00000	02105
1KL03	MJ		0KP00		DENSITY	65033	45	00000	02056
1KL04	MJ		0KR23		SPEED SOUND	65034	45	00000	02043
1KL05	MJ	00000	0KR17		TEMPERATURE	65035	45	00000	02035
1KL06	RJ	0KP32	0KP24		ALL	65036	37	02116	02106
1KL07	TP	A	0KT01			65037	11	20000	00030
1KL08	TP	0KT04	A			65040	11	00033	20000
1KL09	RJ	0KR33	0KR26			65041	37	02055	02046
1KL10	TP	A	0KT03			65042	11	20000	00032
1KL11	MJ		0KL01			65043	45	00000	02001
1KR00	TP	A	0KT05			65044	11	20000	00034
1KR01	TJ	00013	0KR09			65045	42	00015	02025
1KR02	RP	20006	0KR12			65046	75	20006	02030
1KR03	TJ	0KE01	0KR04			65047	42	02120	02020
1KR04	QT	00014	0KR08			65050	51	00016	02024
1KR05	MP	0KF01	0KR08			65051	71	02165	02024
1KR06	AT	0KR11	0KR08			65052	35	02027	02024
1KR07	RP	30005				65053	75	30005	00000
1KR08	TP	0KE07	0KT00			65054	11	02126	00027
1KR09	11	0KF00	75756	BRB	ALARM	65055	11	02164	75756
1KR10	MJ		0KL00			65056	45	00000	02000
1KR11	TP	0KE07	0KT00			65057	11	02126	00027
1KR12	TV	0KR07	0KR15		ALTITUDE	65060	16	02023	02033
1KR13	TP	0KE10	0KT03		OVER 83KM	65061	11	02131	00032
1KR14	TP	0KE11	0KT04		PUT IN	65062	11	02132	00033
1KR15	RP	10003			CONSTANT	65063	75	10003	00000
1KR16	TP	00013	0KT00		VALUES	65064	11	00015	00027
1KR17	TV	0KL11	0KR22		TMPRTR ENTRY	65065	16	02013	02042
1KR18	RJ	0KR07	0KR00			65066	37	02023	02014
1KR19	SP	0KT03	00034			65067	31	00032	00042
1KR20	MM	0KT05	0KT04			65070	25	00034	00033
1KR21	TP	B0000	A0000			65071	11	30000	20000
1KR22	MJ					65072	45	00000	00000
1KR23	TV	0KL11	0KR33		SOUND ENTRY	65073	16	02013	02055

```
1KR24   RJ 0KR22 0KR18                                  65074  37 02042 02036
1KR25   TP A0000 0KT04                                  65075  11 20000 00033
1KR26   AT 0KF05 0KT03                                  65076  35 02171 00032
1KR27   SN 0KF04 00027                                  65077  33 02170 00033
1KR28   DV 0KT03 A0000                                  65100  73 00032 20000
1KR29   AT 0KF02 A0000                                  65101  35 02166 20000
1KR30   LA A0000 00034                                  65102  54 20000 00042
1KR31   MM 0KT03 0KF03                                  65103  25 00032 02167
1KR32   TP B0000 A0000                                  65104  11 30000 20000
1KR33   MJ                                              65105  45 00000 00000
1KP00   TV 0KL11 0KP21                                  65106  16 02013 02103
1KP01   RJ 0KR07 0KR00                                  65107  37 02023 02014
1KP02   SP 0KF06 00035                                  65110  31 02172 00043
1KP03   PM 0KF07 0KT05                                  65111  24 02173 00034
1KP04   TP B0000 Q0000                                  65112  11 30000 10000
1KP05   SP 0KT05 00033                                  65113  31 00034 00041
1KP06   DV Q0000 0KT06          H SCALED 15             65114  73 10000 00036
1KP07   SP 0KT02 00036          A2                      65115  31 00031 00044
1KP08   PM 0KT01 0KT06                                  65116  24 00030 00035
1KP09   PM 0KT00 0KT06                                  65117  24 00027 00035
1KP10   TN B0000 Q0000                                  65120  13 30000 10000
1KP11   QT 0KF09 0KT06          MANTISSA                65121  51 02175 00035
1KP12   QT 0KF08 Q0000          CHARACTERSTC            65122  51 02174 10000
1KP13   SN Q0000 00047                                  65123  33 10000 00057
1KP14   AT 0KP22 0KP19                                  65124  35 02104 02101
1KP15   LQ 0KT06 00010                                  65125  55 00035 00012
1KP16   SP 0KF10 00036                                  65126  31 02176 00044
1KP17   RP 20004 0KP19          EXPONENTIAL             65127  75 20004 02101
1KP18   PM 0KF11 0KT06           POLYNOMIAL             65130  24 02177 00035
1KP19   LA A0000 00000                                  65131  54 20000 00000
1KP20   TP A0000 A0000                                  65132  11 20000 20000
1KP21   MJ                                              65133  45 00000 00000
1KP22   LA A0000 00051                                  65134  54 20000 00063
1KP23   TV 0KL11 0KP32          PRES  ENTRY             65135  16 02013 02116
1KP24   RJ 0KP21 0KP01            DENS                  65136  37 02103 02057
1KP25   TP A0000 0KT02                                  65137  11 20000 00031
1KP26   RJ 0KR22 0KR19              TEMP                65140  37 02042 02037
1KP27   TP A0000 0KT04                                  65141  11 20000 00033
1KP28   MP 0KT04 0KT02                                  65142  71 00033 00031
1KP29   PM 00013 0KF15                                  65143  24 00015 02203
1KP30   LA A0000 00011                                  65144  54 20000 00013
1KP31   TP B0000 A0000                                  65145  11 30000 20000
1KP32   MJ                                              65146  45 00000 00000
1KE00                                                   65147  00 00000 00000
1KE01   03 60891 67000   4  15                         65150  01 06371 12540
1KE02   08 24472 60000   4  15                         65151  02 41017 20510
1KE03   01 55380 10000   5  15                         65152  04 57364 06315
1KE04   01 75261 90000   5  15                         65153  05 26235 71463
1KE05   02 52624 00000   5  15                         65154  07 55320 00000
```

PX 71900-9-(145)

```
1KE06   02 75000 00000    5  15                    65155  10 31070 00000
1KE07   -4 80638 20000       24  A60               65156  77 73143 10362
1KE08   -7 56281 60000  - 5   44  A61               65157  76 60545 34152
1KE09                            A62      ZERO     65160  00 00000 00000
1KE10   03 63400 00000    2  21  B60               65161  00 55331 46315
1KE11                            B61      ZERO     65162  00 00000 00000
1KE12   -1 51932 10000    1  24  A5                65163  77 60635 04711
1KE13   -2 62697 80000  - 6   44  A51               65164  77 75175 51561
1KE14   -1 26083 10000  -10   64  A52               65165  75 65275 41451
1KE15   08 38663 00000    2  21  B50               65166  01 50652 33514
1KE16   -1 88169 00000  - 3   41  B51               65167  74 11271 72771
1KE17   -1 02109 67000    1  24  A4 P              65170  77 65623 77020
1KE18   -5 31594 90000  - 5   44  A41               65171  77 10204 15651
1KE19                            A42      ZERO     65172  00 00000 00000
1KE20   05 08790 00000    2  21  B40               65173  00 77462 43656
1KE21                            B41      ZERO     65174  00 00000 00000
1KE22   -6 74576 40000       24  A3                65175  77 71202 12633
1KE23   -9 41444 00000  - 5   44  A31               65176  76 35220 47027
1KE24   01 20331 60000  -10   64  A32               65177  02 04234 52625
1KE25   02 56039 00000  - 2   21  B30               65200  00 40002 37575
1KE26   01 62733 00000  - 3   41  B31               65201  03 25230 21102
1KE27   -7 96533 90000       24  A2                65202  77 70021 57612
1KE28   -6 93417 20000  - 5   44  A21               65203  76 67224 34510
1KE29                            A22      ZERO     65204  00 00000 00000
1KE30   03 90200 00000    2  21  B20               65205  00 60614 63146
1KE31                            B21      ZERO     65206  00 00000 00000
1KE32   -8 71834 10000       24  A1                65207  77 67220 15315
1KE33   -4 15298 90000  - 5   44  A11               65210  77 24347 63660
1KE34   -1 91052 20000  -10   64  A12               65211  74 55737 13041
1KE35   05 18650 00000    2  21  B10               65212  01 00651 46315
1KE36   -3 56007 10000  - 3   41  B11               65213  70 55277 53367
1KF00   30 01075 65204 B         ALARM TAG          65214  30 01075 65204
1KF01   00 00005 00000 B                            65215  00 00005 00000
1KF02   01 01181 28000    3  21  SOUND              65216  01 76364 01165
1KF03   05 95687 00000  - 1   35    CON              65217  23 03757 05675
1KF04   04 23184 75900    5  15       STA           65220  14 72420 60447
1KF05   04 18200 00000    2  21            NTS      65221  00 64214 63146
1KF06   04 78490 00000  - 8   53  RE RECIP           65222  00 31540 50405
1KF07   01 00000 00000       32                     65223  04 00000 00000
1KF08   77 76000 00000 B                            65224  77 76000 00000
1KF09   00 01777 77777 B                            65225  00 01777 77777
1KF10   05 95618 00000  - 3   24  E4 EXPONEN         65226  00 00003 03130
1KF11   -5 15546 70000  - 2   24  E3  TIAL           65227  77 77745 46517
1KF12   02 38628 42000  - 1   24  E2 POLYNOM         65230  00 00172 13301
1KF13   -6 93007 28000  - 1   24  E1  IAL            65231  77 77235 13422
1KF14   01 00000 00000       24  EO CONSTANTS       65232  00 01000 00000
1KF15   01 19200 00000    1  29  R                  65233  05 75341 21727
```

PX 71900-9-(145)

THE RAMO-WOOLDRIDGE CORPORATION
Los Angeles 45, California

MANUAL INSPECTION AND INSERTION

Designation:	MII-0
Type:	Service Routine
Special Storage:	The constant and temporary storage pools are not used by this program.
Entrances:	Address 40002b for inspection, address 40003b for insertion.
Exit:	Manual stop.

Coded by:	Walter F. Bauer	March 28, 1955
Machine Checked by:	Merritt Elmore	April 1, 1955
Approved by:	Wesley C. Dixon	April 8, 1955

PX 71900-9-(146)

DESCRIPTION

The routine is designed to facilitate manual data read-in and read-out by the machine operator at the control console. With the routine, for example, only four steps are necessary to insert a word rather than approximately eleven without it. -The routine also facilitates the inspection (or insertion) of a succession of words. The information is entered into or read out of the machine by means of the Q-register.

OPERATING INSTRUCTIONS

The following steps should be followed to inspect program data, assuming the computer halted at end of main pulse 6:

1. Set PAK to 40002.
2. Set address n of word to be inspected into right 15 bits of accumulator.
3. Start computer.

The computer will halt with the word displayed in the Q-register. Upon restarting, the word in address n + 1 will be displayed in the Q-register, thus successive starts will cause words in successive storage locations to be displayed. To perform step 2, the computer must be in the "test" mode.

The following steps are taken to enter a word into the memory, assuming the computer halted at the end of main pulse 6:

1. Set PAK to 40003.
2. Set address n of word to be inserted into right 15 bits of accumulator.
3. Enter word to be inserted into quotient register.
4. Start computer.

The computer will halt after inserting the word into address n. Another word, entered into the Q-register, will be inserted into address n + 1 upon starting the computer. That is, upon repeating steps 3 and 4, words are entered into successive memory locations. To perform steps 2 and 3, the computer must be in the "test" mode.

SPECIAL WARNING

This routine does not retain the information which was in the accumulator or quotient register upon its initiation.

THE RAMO-WOOLDRIDGE CORPORATION
Los Angeles 45, California

CENTRAL EXCHANGE SINE-COSINE ROUTINE

Specifications

Identification Tag:	SIN-0
Type:	Subroutine
Assembly Routine Specification:	SUB 49368 04008
Storage:	30 instructions 10 constants in program No words of temporary storage used in program 40 words of total program storage 2 words temporary storage pool used, addresses 00027b thru 00030b
Drum Assignment:	62230b thru 62277b
Entrance and Exit:	RJ SUB01 SUB03 for the Sine RJ SUB01 SUB02 for the Cosine
Alarm:	The alarm exit is not used
Average Machine Time:	3.63 milliseconds for Sine, 3.67 milliseconds for Cosine
Maximum Machine Time:	4.38 milliseconds for Sine, 4.42 milliseconds for Cosine
Mode of Operation:	Fixed point

Coded by:	A. E. Roberts, Jr.	December, 1954
Code Checked by:	Merritt Elmore	April 10, 1955
Machine Checked by:	Merritt Elmore	April 14, 1955
Approved by:	Wesley C. Dixon	May 5, 1955

Description

When supplied with an argument X ($-2^{38} < X < 2^{38}$) scaled by 2^{33} (ie. $X \cdot 2^{33}$) it will compute either $2^{33} \cdot \sin \frac{(\pi X)}{2}$ or $2^{33} \cdot \cos \frac{(\pi X)}{2}$ depending on which of the two possible entrances is used. The Tchebycheff polynomial expansion is used.

The absolute value of the error is less than 2^{-32}.

Programming Instructions

Assume that the subroutine is stored at SUB00 and that X is the angle (in quadrants) whose sine or cosine is desired.

1. Place $X \cdot 2^{33}$ in the accumulator
2. Execute RJ SUB01 SUB03 for the sine, or

   RJ SUB01 SUB02 for the cosine.

Control will be returned to the cell immediately following the return jump with either $2^{33} \cdot \sin \frac{\pi X}{2}$ or $2^{33} \cdot \cos \frac{\pi X}{2}$ left in the accumulator.

Mathematical Method

Method used, $\sin \frac{(\pi X)}{2}$.

Summary. X is transformed into two parts: A two bit integer q designating the quadrant in which the angle lies; and a 33 bit fractional part y which is used in the polynomial approximation for $\sin \frac{(\pi y)}{4}$. $\cos \frac{(\pi y)}{2}$ is obtained from $\sin \frac{(\pi y)}{4}$ using the half angle formula. Finally, $\sin \frac{(\pi X)}{2} = \pm \cos \frac{(\pi y)}{2}$ according to the quadrant.

In Detail

1. Make the argument positive by adding 2^{38} (since $|X| < 2^{38}$).

$$X + 2^{38} = u, \quad 0 \leq u$$

2. The new argument u has an integral part and a fractional part. Throw away all the bits of the integral part except the two immediately to the left of the binary point (since the rest only add integral multiplies of 2π to the angle).

3. Let q = two bit integral part of u. Thus, q is one less than the number of the quadrant in which the angle lies.

4. If X^* is the fractional part of u, then

   a) $y = X^*$ for q = 01 or 11, and

   b) $y = X^* - 1$ for q = 00 or 10, because $\sin \frac{(\pi X)}{2} = \cos \left[\frac{\pi (X-1)}{2}\right]$.

5. $$\sin \frac{(\pi y)}{4} = y\left(\sum_{i=1}^{5} a_{2i+1}\, y^{2i+1}\right), \quad |y| < 1 \text{ from above.}$$

NOTE: This is derived from the Tshebycheff expression

$$\sin \frac{(\pi X)}{4} = 2\sum_{k=1}^{\infty} (-1)^k J_{2k-1}(\pi/4)\left[T_{2k-1}(X)\right].$$

6. $$\cos \frac{(\pi y)}{2} = 1 - 2 \sin^2 \frac{(\pi y)}{4}$$

7. $$\sin \frac{(\pi X)}{2} = \cos \frac{(\pi y)}{2}, \quad q = 00 \text{ or } 01$$

   $$\sin \frac{(\pi X)}{2} = -\cos \frac{(\pi y)}{2}, \quad q = 10 \text{ or } 11$$

Method used, $\cos \frac{(\pi X)}{2}$.

$X + 1$ replaces X and $\sin \frac{(\pi X)}{2}$ is formed as above.

Machine Checking

A driver routine was used to take the sine and cosine of 68 values of $\frac{(\pi X)}{2}$ and the results were checked against values obtained from the National Bureau of Standards Table of Sines and Cosines.

```
D          82S00 49368                                  62230  00 00000 00000
D          81S00 01024                                  02000  00 00000 00000
82S00   FS 00000 00000          NO ALARM XIT            62230  57 00000 00000
82S01   MJ 00000 00000          NORMAL EXIT             62231  45 00000 00000
82S02   AT 81S39 A0000          COSINE ENTRY            62232  35 02047 20000
82S03   LA A0000 00001          SINE ENTRY              62233  54 20000 00001
82S04   AT 00016 Q0000          QAD NUM IN Q            62234  35 00020 10000
82S05   QJ 81S06 81S08          IS QUAD 1  2            62235  44 02006 02010
82S06   TP 81S29 81S28           NO  SET NEG            62236  11 02035 02034
82S07   MJ 00000 81S09           VALUE FUNC             62237  45 00000 02011
82S08   TP 81S26 81S28          YES POS FUNC            62240  11 02032 02034
82S09   QT 81S32 A0000          X STAR IN A             62241  51 02040 20000
82S10   QJ 81S12 81S11          IS QUAD 2  4            62242  44 02014 02013
82S11   SS 81S33 00000           NO  SUB ONE            62243  34 02041 00000
82S12   MP A0000 Q0000          Y SQUARE                62244  71 20000 10000
82S13   SA 81S33 00036           ROUND SCALE            62245  32 02041 00044
82S14   TN A0000 00023           STORE                  62246  13 20000 00027
82S15   TU 81S31 81S19          SET LOOP                62247  15 02037 02023
82S16   TP 81S34 Q0000          A4 TO Q                 62250  11 02042 10000
82S17   MP Q0000 00023           L  CALC                62251  71 10000 00027
82S18   SA 81S33 00036           O  PPLY                62252  32 02041 00044
82S19   AT       Q               O   NOMIAL            62253  35 00000 10000
82S20   RA 81S19 00015           P  UP INDEX            62254  21 02023 00017
82S21   TJ 81S30 81S17               TEST END           62255  42 02036 02021
82S22   MP Q0000 Q0000          CALCULATE               62256  71 10000 10000
82S23   SA 81S33 00036                                  62257  32 02041 00044
82S24   MP A0000 00023            NEG COS OF            62260  71 20000 00027
82S25   SS 81S39 00038            ONE HALF              62261  34 02047 00046
82S26   TN A0000 A0000            PI Y                  62262  13 20000 20000
82S27   ST 81S39 A0000                                  62263  36 02047 20000
82S28   00 00000 00000          POS OR NEG              62264  00 00000 00000
82S29   MJ 00000 81S01          JUMP TO EXIT            62265  45 00000 02001
82S30   AT 81S39 Q0000                                  62266  35 02047 10000
82S31   FS 81S35 00000          LOOP U ADDRS            62267  57 02043 00000
82S32   37 77777 77774 B        MASK                    62270  37 77777 77774
82S33   40 00000 00000 B        ROUND OFF 36            62271  40 00000 00000
82S34   02 71415 00000    6     A4                      62272  00 00122 65046
82S35   08 04213 96000    7     A3                      62273  00 04626 21024
82S36   01 36910 77440    9     A2                      62274  01 21465 66440
82S37   01 10975 78641   10     A1                      62275  12 25357 16221
82S38   02 69860 75408   10     A                       62276  31 10375 52420
82S39   01 00000 00000       33 ONE                     62277  10 00000 00000
```

PX 71900-9-(147)

THE RAMO-WOOLDRIDGE CORPORATION
Los Angeles 45, California

Specifications

Identification Tag:	SIN-1
Type:	Subroutine
Assembly Routine Specification:	SUB 49420 04210
Storage:	32 instructions, 10 constants in program, 42 total program storage. 2 words of temporary storage pool used, addresses 27b and 30b
Entrance and Exit:	RJ 00K01 00K03 for the sine, or RJ 00K01 00K02 for the cosine.
Alarm:	The alarm exit is not used but a divide fault may occur.
Drum Assignment:	62314b thru 62365b
Average Machine Time:	2.6 milliseconds
Maximum Machine Time:	4.0 milliseconds (estimated)
Mode of Operation:	Fixed point

Coded by:	M. Elmore	March 21, 1955
Code Checked by:	R. Beach	March 23, 1955
Machine Checked by:	M. Elmore	April 26, 1955
Approved by:	W. Dixon	May 11, 1955

Description

When supplied with an argument $x(-2\pi \cdot 2^{32} \leq x \leq 2\pi \cdot 2^{32})$ scaled by 2^{32} (i.e. $x \cdot 2^{32}$) this routine will compute either $2^{32} \cdot \sin x$ or $2^{32} \cdot \cos x$, depending on which of two entrances is used. Computation is by means of the Rand approximation using the repeat polynomial multiply instruction. (Hence, SIN-1 should not be used on 1103 computers not equipped with the polynomial multiply).

The absolute error is less than 2^{-27}.

Programming and Operating Instructions

1. Place $x \cdot 2^{32}$ in A.

2. RJ to the subroutine

   If the subroutine has been assigned some arbitrary region, say 00K00, then the address of the sine entrance is 00K03, the address of the cosine entrance is 00K02, and the address of the exit is 00K01. One should, therefore, use one of two RJ instructions to enter the routine.

   ```
   RJ   00K01   00K03    to obtain the sine, or
   RJ   00K01   00K02    to obtain the cosine.
   ```

3. At the time of exit from the routine either $2^{32} \cdot \sin x$ or $2^{32} \cdot \cos x$ will be left in A.

4. A divide fault occurs if x exceeds the range (see note in Mathematical Analysis section).

Mathematical Analysis

Method used, sin x.

Summary. First x is reduced to a positive angle less than 2π; second it is transformed into an angle with an absolute value less than or equal to $\pi/2$, and the sign is assigned according to the quadrant in which the angle lies. Calculation then proceeds using the Rand approximation.

In Detail

1. (a) If x is positive and greater than 2π, 2π is subtracted from x until x is negative, when 2π is added back in.

   (b) If x is negative, 2π is added to x until x is positive.

NOTE: The number of times that x and $\pm 2\pi$ are added is tallied with a Q-jump. If positive x does not become negative or negative x does not become positive after six subtractions or additions respectively, x is divided by 2π and the remainder becomes x.

Because of this division, if $|x| \geq 2\pi (2^{32}+6)$, a divide fault will result.

2. If x is greater than or equal to $\pi/2$, π is subtracted until x is less than $\pi/2$. If this takes only one subtraction, then the negative of the newly obtained x is used for x. If it takes two subtractions, no further change of sign is made.

3. $\sin x = \sum_{i=1}^{5} a_{2i-1} x^{2i-1}$, $\qquad \frac{\pi}{2} \leq x < \frac{\pi}{2}$, from above.

This polynomial is related to the Rand Sheet No. 16 polynomial by the following relation:

$$a_k = C_k \left(\frac{2}{\pi}\right)^k$$

Method used, cos x.

x is replaced by $x + \pi/2$ if x is negative or by $x-3\pi/2$ if x is positive, and sin x is found as above.

Machine Checking

The sin x and cos x of 34 values of x were found and the results checked against values for the same angles found using the SIN-0 routine. The greatest error was $45 \cdot 2^{-33}$ (approximately $.71 \cdot 2^{-27}$).

```
D          82S00 49420                                    62314  00 00000 00000
D          81S00 01024                                    02000  00 00000 00000
82S00   FS 00000 00000                                    62314  57 00000 00000
82S01   MJ 00000 00000                                    62315  45 00000 00000
82S02   SJ 81S09 81S08          COSINE ENTRY              62316  46 02011 02010
82S03   TP 81S41 Q0000          SINE ENTRY                62317  11 02051 10000
82S04   TJ 81S32 81S15          IS X SML NUF              62320  42 02040 02017
82S05   QJ 81S06 81S11          TLLY SUB 2PI              62321  44 02006 02013
82S06   DV 81S32 Q0000          DIV BY 2PI                62322  73 02040 10000
82S07   SJ 81S14 81S18                                    62323  46 02016 02022
82S08   ST 81S32 A0000           SUB 2PI                 62324  36 02040 20000
82S09   AT 81S33 A0000           ADD HALF PI             62325  35 02041 20000
82S10   MJ 00000 81S03                                    62326  45 00000 02003
82S11   ST 81S32 A0000          SUB 2PI TIL               62327  36 02040 20000
82S12   SJ 81S14 81S05           X GOES NEG              62330  46 02016 02005
82S13   QJ 81S06 81S14          TLLY ADD 2PI              62331  44 02006 02016
82S14   AT 81S32 A0000          ADD 2PI TIL               62332  35 02040 20000
82S15   SJ 81S13 81S18           X GOES POS              62333  46 02015 02022
82S16   TN A      A             CHNGE SIGN X              62334  13 20000 20000
82S17   MJ        81S22         JMP TO POLYN              62335  45 00000 02026
82S18   TJ 81S33 81S22          IS X 1ST QUD              62336  42 02041 02026
82S19   ST 81S34 A               NO   SUB PI             62337  36 02042 20000
82S20   TJ 81S33 81S16          IS X SML NUF              62340  42 02041 02020
82S21   ST 81S34 A               NO   SUB PI             62341  36 02042 20000
82S22   TP A        23          SAVE X                    62342  11 20000 00027
82S23   MP    23 Q              X SQUARE                  62343  71 00027 10000
82S24   SA 81S40    41          ROUND SCALE               62344  32 02050 00051
82S25   TP A        24           STORE X SQ               62345  11 20000 00030
82S26   SP 81S35    35          C9 TO B                   62346  31 02043 00043
82S27   RP 2   4 81S29          COMPUTE                   62347  75 20004 02035
82S28   PM 81S36    24           SIN X                    62350  24 02044 00030
82S29   PM    13    23           POLYNOMIAL               62351  24 00015 00027
82S30   TP B      A             SCALE ANSWER              62352  11 30000 20000
82S31   MJ        81S01         JUMP TO EXIT              62353  45 00000 02001
82S32   06 28318 53072       32 2PI                       62354  31 10375 52421
82S33   01 57079 63268       32 ONE HALF PI               62355  06 22077 32504
82S34   03 14159 26536       32 PI                        62356  14 44176 65211
82S35   02 60188 69075 - 6   43 C9 SCALED 43              62357  00 01272 34047
82S36   -1 98074 14309 - 4   40 C7 SCALED 4               62360  77 63011 57010
82S37   08 33302 51737 - 3   38 C5 SCALED 38              62361  02 10416 36646
82S38   -1 66666 56696 - 1   36 C3 SCALED 36              62362  65 25252 70030
82S39   09 99999 99470 - 1   34 C1 SCALED 34              62363  17 77777 77645
82S40   01                B     ROUND OFF                 62364  01 00000 00000
82S41   00 70000 00000 B                                  62365  00 70000 00000
```

PX 71900-9-448)

THE RAMO-WOOLDRIDGE CORPORATION
Los Angeles 45, California

Small Angle Sine-Cosine Routine

Specifications

Identification Tag:	SIN-2
Type:	Subroutine
Assembly Routine Spec:	SUB 49770 03009
Storage:	21 instructions
	9 constants in program
	30 words total program storage
	2 words temporary storage pool used, addresses 00027b and 00030b
Drum Assignment:	Addresses 63052b thru 63107b
Entrance and Exit:	RJ 00K01 00K03 for the sine
	RJ 00K01 00K02 for the cosine
Alarm:	The alarm exit is used to print "SIN-2" if the argument is too large
Machine Time:	2.20 ms average, 3.14 ms maximum
Mode of Operation:	Fixed point

Coded by:	M. Elmore	April 22, 1955
Machine Checked by:	M. Elmore	April 29, 1955
Approved by:	W. Dixon	July 20, 1955

Description

When supplied with an argument x $(-\pi/2 \leq x < \pi/2)$ scaled by 2^{32} (that is, $x \cdot 2^{32}$) this routine will compute either $2^{32} \cdot \sin x$ or $2^{32} \cdot \cos x$, depending on which of two entrances is used. Computation is by means of the Rand approximations using the repeated polynomial multiply instruction (hence, SIN-2 should not be used on 1103 computers not equipped with the polynomial multiply instruction).

The absolute error is less than 2^{-27}.

Programming Instructions

1. Place $x \cdot 2^{32}$ in A

2. RJ to the subroutine. If the subroutine has been assigned some arbitrary region, say 00K00, then the address of the sine entrance is 00K03, the address of the cosine entrance is 00K02, and the address of the exit is 00K01. One should therefore use one of two RJ instructions to enter the routine:

   RJ 00K01 00K03 to obtain the sine, or
   RJ 00K01 00K02 to obtain the cosine.

3. At the time of exit from the routine, either $2^{32} \cdot \sin x$ or $2^{32} \cdot \cos x$ will be left in A.

4. An alarm occurs if x exceeds the range

$$\pi/2 \leq x < \pi/2$$

Mathematical Analysis

Method used for sin x:

$$\sin x = \sum_{i=1}^{5} a_{2i-1} \quad x^{2i-1}$$

This polynomial is related to the Rand Sheet No. 16 polynomial by the following relation:

$$a_K = c_K \left(\frac{2}{\pi}\right)^K.$$

Method used for cos x:

x is replaced by $(\pi/2) - |x|$ and sin x is found as above.

Machine Checking

The sin x and cos x of 15 values of x were found and the results were checked against results of SIN-1 for the same angles. The answers were identical, which is to be expected since the same polynomial was used in both routines.

```
D          82S00 49770                                  63052  00 00000 00000
D          81S00 01024                                  02000  00 00000 00000
82S00   37 75701 75702 B                                63052  37 75701 75702
82S01   MJ 00000 00000                                  63053  45 00000 00000
82S02   TJ 81S21 81S06          COSINE ENTRY            63054  42 02025 02006
82S03   TJ 81S21 81S10          SINE ENTRY              63055  42 02025 02012
82S04   11 81S23 75756 BRB      ALARM TAG               63056  11 02027 75756
82S05   MJ 00000 81S00           TO ALR                 63057  45 00000 02000
82S06   TJ 81S22 81S04                                  63060  42 02026 02004
82S07   TM A0000 00023          HALF PI                 63061  12 20000 00027
82S08   TP 81S21 A0000           MINUS X                63062  11 02025 20000
82S09   ST 00023 A0000           FOR COS X              63063  36 00027 20000
82S10   TJ 81S22 81S04                                  63064  42 02026 02004
82S11   TP A0000 00023          STORE X                 63065  11 20000 00027
82S12   MP 00023 Q0000          X SQUARE                63066  71 00027 10000
82S13   SA 81S24 00041          ROUND SCALE             63067  32 02030 00051
82S14   TP A0000 00024           STORE X SQ             63070  11 20000 00030
82S15   SP 81S25 00035          C9 TO B REG             63071  31 02031 00043
82S16   RP 20004 81S18          COMPUTE                 63072  75 20004 02022
82S17   PM 81S26 00024           SIN X                  63073  24 02032 00030
82S18   PM 00013 00023           POLYNOMIAL             63074  24 00015 00027
82S19   TP B0000 A0000          SCALE ANSWER            63075  11 30000 20000
82S20   MJ 00000 81S01          JUMP TO EXIT            63076  45 00000 02001
82S21   01 57079 63268       32 ONE HALF PI             63077  06 22077 32504
82S22   -1 57079 63268       32                         63100  71 55700 45273
82S23   24 14065 67404 B        ALARM TAG              63101  24 14065 67404
82S24   01 00000 00000 B        ROUND OFF              63102  01 00000 00000
82S25   02 60188 69075 - 6   43 C9                     63103  00 01272 34047
82S26   -1 98074 14309 - 4   40 C7                     63104  77 63011 57010
82S27   08 33302 51737 - 3   38 C5                     63105  02 10416 36646
82S28   -1 66666 56696 - 1   36 C3                     63106  65 25252 70030
82S29   09 99999 99470 - 1   34 C1                     63107  17 77777 77645
```

PX 71900-9-(149)

THE RAMO-WOOLDRIDGE CORPORATION
Los Angeles 45, California

SQUARE ROOT

Specifications

Identification Tag:	SQR-0
Type:	Subroutine
Assembly Routine Spec:	SUB 49312 04409
Storage:	34 instructions
	10 constants in program
	44 words of total program storage
	4 words of temporary storage pool used, addresses 00027b thru 00032b
Entrance:	Address SUB02
Exit:	Address SUB01
Alarm Exit:	The alarm exit is used.
Machine Time:	2.38 ms. maximum machine time
Mode of Operation:	Fixed point

Coded by:	A. Roberts, Jr. (ERA)	December 1, 1954
Code Checked by:	T. Tack	March 15, 1955
Machine Checked by:	M. Elmore	March 19, 1955
Approved by:	W. Dixon	April 29, 1955

PX 71900-9-(150)

Description

This routine uses a rational function approximation followed by one step of the Newton-Raphson procedure to compute the square root. It was written by A Roberts of E. R. A. and adopted for use at Ramo-Wooldridge.

Programming Instructions

Assume the routine is stored at SUB00. If x is the number whose square root is desired, place $x \cdot 2^{33}$ in the accumulator. Since the entire 72 bits of the accumulator are used in taking the square root the permissible range on x is

$$0 \leq x < 2^{38}$$

Execute the return jump instruction

RJ SUB01 SUB02

Upon exit from the routine the square root of x, scaled by 2^{33} is left in the accumulator.

Alarm Condition

If $x \cdot 2^{33} < 0$ an alarm print of "SQR-0" will occur and the computer will halt with

$$(A) = x \cdot 2^{33}$$

Pushing the start button will return control to the exit of SQR-0.

Mathematicl Method and Accuracy

A first approximation of the form

$$c_1(x+c_4)+c_2 - \frac{c_3}{(x+c_4)}$$

is computed. One application of the Newton-Raphson formula to this first approximation gives the desired square root. 33 bit accuracy was obtained in all cases tested.

```
D          S2S00 49312                                  62140  00 00000 00000
D          01S00 01024                                  02000  00 00000 00000
D          01T00 00023                                  00027  00 00000 00000
D          01A00 00013                                  00015  00 00000 00000
S2S00   37 75701 75702 B                                62140  37 75701 75702
S2S01   MJ 00000 00000       EXIT                       62141  45 00000 00000
S2S02   SJ 01S32 01S03       ARG NEG                    62142  46 02040 02003
S2S03   ZJ 01S04 01S01       ARG ZERO                   62143  47 02004 02001
S2S04   SF A0000 01T00                                  62144  74 20000 00027
S2S05   TP A0000 A0000       CLEAR B                    62145  11 20000 20000
S2S06   TP A0000 01T01       STORE SC ARG               62146  11 20000 00030
S2S07   SA 01S35 00054                                  62147  32 02043 00066
S2S08   TP A0000 01T02                                  62150  11 20000 00031
S2S09   MP 01S36 01T02                                  62151  71 02044 00031
S2S10   AT 01S37 01T03                                  62152  35 02045 00032
S2S11   SN 01S38 00015                                  62153  33 02046 00017
S2S12   DV 01T02 A0000                                  62154  73 00031 20000
S2S13   AT 01T03 01T02                                  62155  35 00032 00031
S2S14   SP 01T01 00032                                  62156  31 00030 00040
S2S15   SS 01T02 00000                                  62157  34 00031 00000
S2S16   DV 01T02 A0000                                  62160  73 00031 20000
S2S17   AT 01T02 01T02                                  62161  35 00031 00031
S2S18   LQ 01T00 00035                                  62162  55 00027 00043
S2S19   QT 01A01 A0000                                  62163  51 00016 20000
S2S20   TV A0000 01S28                                  62164  16 20000 02034
S2S21   TP 01S31 01S29                                  62165  11 02037 02035
S2S22   TJ 01S39 01S24                                  62166  42 02047 02030
S2S23   TP 01S34 01S29                                  62167  11 02042 02035
S2S24   QJ 01S28 01S25                                  62170  44 02034 02031
S2S25   MP 01S40 01T02                                  62171  71 02050 00031
S2S26   SA 01S41 00037                                  62172  32 02051 00045
S2S27   TP A0000 01T02                                  62173  11 20000 00031
S2S28   SP 01T02 00000                                  62174  31 00031 00000
S2S29   00 00000 00000 B                                62175  00 00000 00000
S2S30   TP A0000 A0000                                  62176  11 20000 20000
S2S31   MJ 00000 01S01                                  62177  45 00000 02001
S2S32   11 01S42 75756 BRB                              62200  11 02052 75756
S2S33   MJ 00000 01S00                                  62201  45 00000 02000
S2S34   32 01S43 00044 BRB                              62202  32 02053 00044
S2S35   26 47670 31361 B     D                          62203  26 47670 31361
S2S36   00 00000 65324 B     A                          62204  00 00000 65324
S2S37   11 45346 44516 B     B                          62205  11 45346 44516
S2S38   33 06571 40273 B     C                          62206  33 06571 40273
S2S39   00 00000 00023 B                                62207  00 00000 00023
S2S40   26 50117 14640 B                                62210  26 50117 14640
S2S41   20 00000 00000 B                                62211  20 00000 00000
S2S42   24 35125 63704 B     TAG SQR                    62212  24 35125 63704
S2S43   37 77777 77777 B                                62213  37 77777 77777
```

PX 71900-9-(150)

THE RAMO-WOOLDRIDGE CORPORATION
Los Angeles 45, California

Normally Distributed Pseudo Random Numbers

Specifications

Identification Tag:	RAN-0
Type:	Subroutine
Assembly Routine Spec:	SUB 50010 02007
Storage:	20 words total program storage 2 words temporary pool used, addresses 30b and 31b The constant pool is used.
Entrance and Exit:	RJ SUB00 SUB02 normal entrance RJ SUB00 SUB01 reset entrance
Machine Time:	472 + 690 n microseconds (for definition of n, see Mathematical Method). If n = 6 (the normal case) this gives 4612 microseconds.
Mode of Operation:	Fixed point

Coded by:	R. Bigelow	July, 1955
Code Checked by:	M. Perry	July, 1955
Machine Checked by:	R. Bigelow	August, 1955
Report Written by:	F. Meek	April, 1956
Approved by:	W. Bauer	May, 1956

PX 71900-9-(151)

Description

Repeated use of this subroutine produces a sequence of pseudo random numbers from an approximately normal distribution having mean equal to zero, and standard deviation equal to one.

Programming Instructions

Each entry to this subroutine produces in the accumulator a quantity $x.2^{30}$ where x is a pseudo random number selected from the sequence described above. To obtain each x in the sequence use RJ SUB00 SUB02. The subroutine must not be destroyed between successive entries if the sequence of x's is to have the desired randomness.

To restart the sequence, i.e., to obtain the first x again after the subroutine has been entered one or more times, use RJ SUB00 SUB01.

The subroutine may be modified to cause the sequence of x's to more closely approximate a normal distribution. To make this modification choose a number n such that $5 \leq n \leq 32$ and make (SUB18) = $(n - 1)\cdot 2^{0}$ and make (SUB19) = $(\sqrt{3/n})\cdot 2^{35}$. Increasing n improves the approximation to a normal distribution and increases the machine time (see page 1). The unmodified routine uses n = 6.

Mathematical Method

Let $y_i = 5^{2q+1} y_{i-1} \pmod{2^s}$, where q is a non-negative integer, and y_0 is any odd positive integer less than 2^s (see appendix), define a sequence $\{y_i\}$ of positive integers. Such sequences satisfy many of the properties associated with sequences of random integers uniformly distributed between zero and 2^s. In fact, many such sequences have been computed and tested for "randomness" and for "goodness of fit" to a uniform distribution and have been found satisfactory for most purposes (see Testing).

PX 71900-9-(151)

If one assumes that the sequence $\{y_i\}$ defined above is a sequence of truly random integers uniformly distributed between zero and 2^s, one can obtain a sequence $\{x_j\}$ of random numbers from a distribution which approximates the standard (zero mean and unit standard deviation) normal distribution by appealing to a well-known theorem concerning the distribution of means. On page 69 of [5] there appears the following

"Theorem: If y has a distribution with mean m and standard deviation σ for which the moment-generating function exists, then the variable

1) $$x = (\bar{y}-m)\sqrt{n}/\sigma$$

has a distribution which approaches the standard normal distribution as n becomes infinite". In 1), $\bar{y}$ denotes the average value of n sample values of y. This theorem is applicable to the integers y_i from the sequence $\{y_i\}$ because there certainly exists a moment-generating function* for a uniform distribution. The mean m of the y_i is clearly

2) $$m = \frac{1}{2}\cdot 2^s = 2^{s-1}.$$

The standard deviation σ is easily found to be

3) $$\sigma = \frac{1}{\sqrt{3}}\cdot 2^{s-1} = \frac{m}{\sqrt{3}}.$$

(See the appendix for a derivation of 3).)

Let

4) $$\bar{y}_j = \frac{1}{n}\sum_{k=0}^{n-1} y_{nj+k}$$

denote the average of n consecutive values of y_i from the sequence $\{y_i\}$ beginning with y_{nj}, j=0, 1,2,..., and write 1) as

* For the definition of a moment-generating function, see page 26 of [5].

5) $$x_j = (\bar{y}_j - m)\ \sqrt{n}/\sigma .$$

According to the theorem, x_j is the j^{th} number in a sequence $\{x_j\}$ of random numbers from a distribution which approaches the standard normal distribution as n becomes infinite. On using 2) and 3), 5) becomes

6) $$x_j = \sqrt{3n}\ (\bar{y}_j \cdot 2^{1-s} - 1).$$

In regard to the sequence $\{y_i\}$ defined above, it can be shown that such a sequence has period 2^{s-2}. To obtain the longest period possible, s was chosen to be 35 for this subroutine.

Other computing installations have chosen q as large as possible, subject only to the restriction that 5^{2q+1} must be contained in one storage register. Since multiplication time on the 1103 is a direct function of the number of binary ones in one of the multiplicands, 5^{2q+1} was somewhat arbitrarily chosen to be 5^5. See [1]. In order not to introduce a bias at the beginning of the sequence, the subroutine uses for y_0 the 17th integer in the sequence obtained using $y_0 = 1$, namely, 13 41437 54765b. See [1] .

It was stated under Programming Instructions that the unmodified subroutine uses n = 6. It is believed that this value of n will yield numbers sufficiently "normal" for most purposes. See Testing.

The subroutine does not use formulas 4) and 6) to compute x_j. Instead the following equivalent process is used: Form

7) $$\sum_{k=0}^{n-1} \left[(y_{nj+k} - 2^{34}) \cdot 2^{-4} \right] = z_j \cdot 2^{30} ,$$

and then,

8) $$x_j \cdot 2^{30} = \frac{1}{\sqrt{2}} z_j \cdot 2^{30} .$$

Using n = 6, and s = 35, it is easy to verify that 7) and 8) are equivalent to 4) and 6).

Testing

In the first paragraph under Mathematical Method it was stated that many sequences $\{y_i\}$ have been computed and tested for "randomness" and for "goodness of fit" to a uniform distribution. One such testing program was carried out on the SEAC [2] using 2q+1 = 17 and s = 42. Another testing program was carried out on the ORDVAC [3] using 2q+1 = 13 and s = 39. A similar process for generating pseudo-random integers on a decimal computer, i.e., the UNIVAC, was devised and tested [4]. In all of these testing programs, the integers generated were judged to be "satisfactory". But in all these testing programs q (or its analog on the UNIVAC) was chosen as large as possible, subject only to the particular machine register capacity. Therefore, the small value of q used by this subroutine on the 1103 (in order to save multiplication time) is somewhat questionable. In fact, Juncosa [3] states that if 5^k is used, k should be an odd integer such that 5^k is "preferably slightly less than 2^s". The choice of n = 6 is likewise questionable. Therefore, it was decided to write a routine, which shall hereinafter be referred to as the testing routine, which would in some sense test the pseudo normally distributed random numbers x_j. The results obtained with the testing routine can be termed satisfactory. For more details, see the appendix.

The subroutine, with q = 2 and n = 6, has already been used with apparently satisfactory results. It was used to evaluate a function in the form of a Fourier series with normally distributed random coefficients. It was also used to simulate

random and correlated radar noise. In this connection, the theoretical distribution of x_j was computed and plotted* (assuming the y_i to be from a truly uniform distribution) with n = 6. This plot is shown, along with the standard normal distribution, as figure 4 in the appendix. The difference between the two curves would be considered negligible for most applications. However, there is a difference which does not show on the figure, and which might be considered important for some applications. From 4) and 6) under Mathematical Method, one can show that

$$|x_j| < \sqrt{3n} < 4.25$$

for n = 6. Hence the probability of obtaining an $|x_j| \geq 4.25$ is zero. But, for the standard normal distribution, the probability of obtaining an $|x| \geq 4.25$ is about one in fifty thousand.

---

* The writer is indebted to R. Schwarz for suggesting the computation and carrying it out.

References

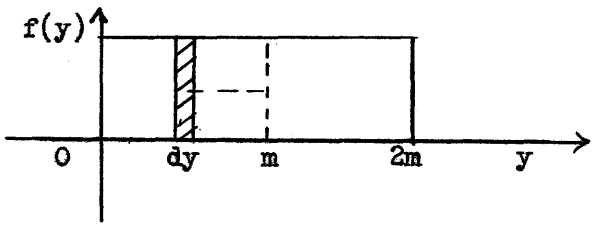
1. Pseudo-Random Number Generator Subroutine, CV-28, November, 1954, 1103 Central Exchange Newsletter Number 3, January, 1955.
2. Generation of Pseudo-random Numbers, NBS Report 3370, Olga Taussky and John Todd, June 22, 1954.
3. Random Number Generation on the BRL High-Speed Computing Machines, BRL Report No. 855, M. L. Juncosa, May, 1953.
4. The Generation of Pseudo-Random Numbers on a Decimal Calculator, ACM Journal, vol. 1, No. 2, pp. 88-91, Jack Moshman, April, 1954.
5. Paul G. Hoel, Introduction to Mathematical Statistics, John Wiley and Sons, New York, 1947.

APPENDIX

Derivation of σ

If the uniform distribution function f(y) is chosen so that the area under the curve is unity, the variance σ^2 is equal to the second moment about the mean m, i.e.,

$$\sigma^2 = \int_0^{2m} (y-m)^2 f(y)\, dy,$$



where $2m = 2^s$, $m = \frac{1}{2} \cdot 2^s = 2^{s-1}$. To make the area equal to unity, f(y) must be 1/2m. Thus

$$\sigma^2 = 1/2m \int_0^{2m} (y^2 - 2my + m^2)\, dy,$$

or $\sigma^2 = \frac{m^2}{3}$, and $\sigma = \frac{m}{\sqrt{3}}$ which is 3) under Mathematical Method.

The Testing Routine

The testing routine uses RAN-0 (with q = 2 and n = 6) to generate in sequence any number M of sets of N x_j's, likewise in sequence. For example, the data used in figures 1, 2, and 3, was produced using M = 512 and N = 100, so that MN = 51,200 x_j's were generated in sequence and divided into 512 sets of 100 each. For each set of N x_j's the following three quantities are computed and punched on cards using CPO-2:

1. $\bar{x} = \frac{1}{N} \sum_{j=1}^{N} x_j$.

2. $$s^2 = \frac{1}{N} \sum_{j=1}^{N} (x_j^2 - \bar{x}^2) .$$

$\bar{x}$ is the sample mean and s^2 is the sample variance.

3. $$\chi^{2*} = \sum_{i=1}^{10} \frac{(\theta_i - e_i)^2}{e_i}$$

where θ_i is the actual number of x_j's for which $a_i \leq x_j < a_{i+1}$, and e_i is the number of x_j's for which it is expected that $a_i \leq x_j < a_{i+1}$, assuming that the x_j's are distributed normally with zero mean and unit standard deviation. The a_i are given in the table below along with the corresponding e_i/N. The values of e_i/N are from Table II, pp. 243 - 245 of [5].

i	a_i	e_i/N	i	a_i	e_i/N
1	$-\infty$	.02275	6	0	.19146
2	-2.0	.04406	7	0.5	.14988
3	-1.5	.09185	8	1.0	.09185
4	-1.0	.14988	9	1.5	.04406
5	-0.5	.19146	10	2.0	.02275
			11	$+\infty$	0

χ^2 is a measure of the "goodness of fit" of the generated x_j to the standard normal distribution.

On figures 1, 2, and 3 are plotted the histograms of the actual frequencies obtained from M = 512 sets of N = 100 x_j's each. Figures 1, 2, and 3 refer respectively to the computed quantities $\bar{x}$, s^2, and χ^2. On each figure is also plotted the theoretical distribution curve of the appropriate quantity.

* Chi - squared

If the x_j are normally distributed with $m_x = 0$ and $\sigma_x = 1$, $\bar{x}$ will also be normally distributed with $m_{\bar{x}} = 0$ and $\sigma_{\bar{x}} = \sigma_x/\sqrt{N} = .1$.

At the top of page 138 of [5] there appears the following Theorem: If x is normally distributed with variance σ^2, and s^2 is the sample variance based on a random sample of size N, then Ns^2/σ^2 has a χ^2 distribution with N - 1 degrees of freedom.

The above theorem is applicable with $\sigma = 1$ and N = 100, and using (8), page 134 of [5] , for the distribution function of χ^2, one obtains

$$f(100s^2/_1) = f(100s^2) = f(u^2) = \frac{1}{2^{99/2}\Gamma(\frac{99}{2})} (u^2)^{97/2} e^{-u^2/2}$$

which reduces to

$$f(u^2) = \frac{1}{2\Gamma(\frac{99}{2})} (\frac{u^2}{2})^{97/2} e^{-u^2/2} ,$$

where $u^2 = 100s^2$. This latter expression for $f(u^2)$ is rather easily evaluated using natural logarithms, i.e.,

$$\log f(u^2) = 97/_2 \log (\frac{u^2}{2}) - \frac{u^2}{2} - \log 2 - \log\Gamma(\frac{99}{2}).$$

$\log \Gamma(\frac{99}{2})$ can be approximated using Stirling's formula or it can be evaluated directly using

$$\log \Gamma(\frac{99}{2}) = \log \frac{97}{2} + \log \frac{95}{2} + \log \frac{93}{2} + \ldots + \log \frac{3}{2} + \log \Gamma(\frac{3}{2}).$$

Either method yields $\log \Gamma(\frac{99}{2}) \sim 142.6172$. The expression above for $\log f(u^2)$ was used to obtain a sufficient number of points to plot the theoretical distribution curve of s^2 on figure 2.

The theoretical distribution curve of χ^2 with 9 degrees of freedom which is plotted on figure 3, was obtained from data given in Table III, page 246 of [5] .

(The number of degrees of freedom is one less then the number of groupings used to compute χ^2 for a sample of size N.)

For each of the three figures the frequency histograms and the theoretical distribution curves have been adjusted to have the same area under them.

Remarks on the Figures, Randomness, and the Choice of q

For all three figures the histograms seem to follow the curves fairly well. This indicates that the x_j generated by the subroutine (with q = 2 and n = 6) are distributed in a fashion which approximates the standard normal distribution to a fairly high degree. Figure 4 suggests the same thing but it is entirely theoretical. It should be emphasized that none of the figures shows anything about the randomness of the x_j. An inspection of the data which produced the histograms indicates that the computed $\bar{x}$, s^2, and χ^2 for consecutive sets of 100 x_j's are random. But again, this says nothing for the randomness of the x_j's themselves. The testing of the randomness of the x_j's was deliberately avoided because it was assumed from the results of the testing done on the sequences of the pseudo-uniformly distributed integers by other computer installations, that the sequence $\{y_i\}$ with q = 2, and hence the sequence $\{x_j\}$ computed by the subroutine, would be sufficiently random for our purposes. As was pointed out under Testing, the only significant difference between the sequence $\{y_i\}$ computed by the subroutine and those sequences tested at other installations was the value of q. For the subroutine q was chosen small so as to shorten the multiplication time on the 1103. The other installations chose q as large as possible because it was intuitively felt that such a choice might produce more randomness, such a choice would automatically eliminate any bias at the beginning of the sequence with $y_0 = 1$, and the computer multiplication times did not depend on q. But since the period of the left-

most binary digit of the integers y_i, and hence the period of the sequence $\{y_i\}$, is 2^{s-2}, independent of q ([2] and [4]), and since it is believed that the randomness and the uniformity of the sequence $\{y_i\}$ is due to this long period (s being chosen as large as possible), it appears also that the randomness and uniformity of the sequence $\{y_i\}$ is independent of q.

Remarks on the Choice of y_0

It was stated in the first paragraph under Mathematical Method that y_0 can be any odd positive integer less than 2^s. There are 2^{s-1} such integers and they can be divided into 2 groups of 2^{s-2} integers each, according to whether, in binary representation, they end in 01 or 11, i.e., whether they are of the form $4k + 1$ or $4k + 3$, k being any non-negative integer less than 2^{s-2}. Now let $s > 2$, $q \geq 0$, s and q such that $5^{2q+1} < 2^s$. If $y_0 = 4k + 1 < 2^s$, e.g., $y_0 = 1$, the first 2^{s-2} integers y_i of the sequence $\{y_i\}$ will be the 2^{s-2} distinct integers of the form $4k + 1$, and then the sequence will repeat itself. Similarly, if $y_0 = 4k + 3 < 2^s$, e.g., $y_0 = 3$, the 2^{s-2} distinct integers generated will be of the form $4k + 3$.

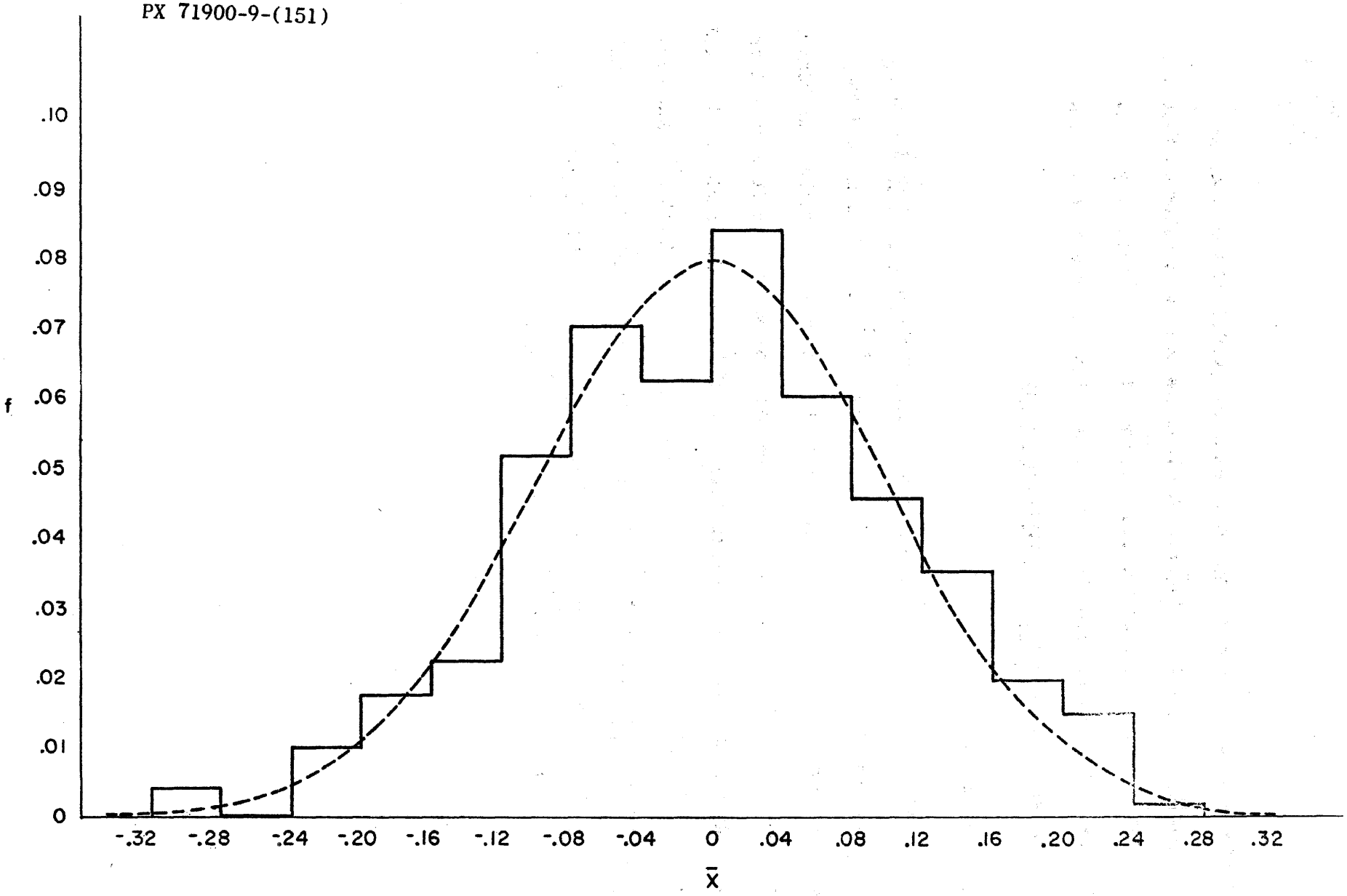

.10
.09
.08
.07
f .06
.05
.04
.03
.02
.01
0
-.32 -.28 -.24 -.20 -.16 -.12 -.08 -.04 0 .04 .08 .12 .16 .20 .24 .28 .32
$\bar{X}$

FIG. 1

PX 71900-9-151

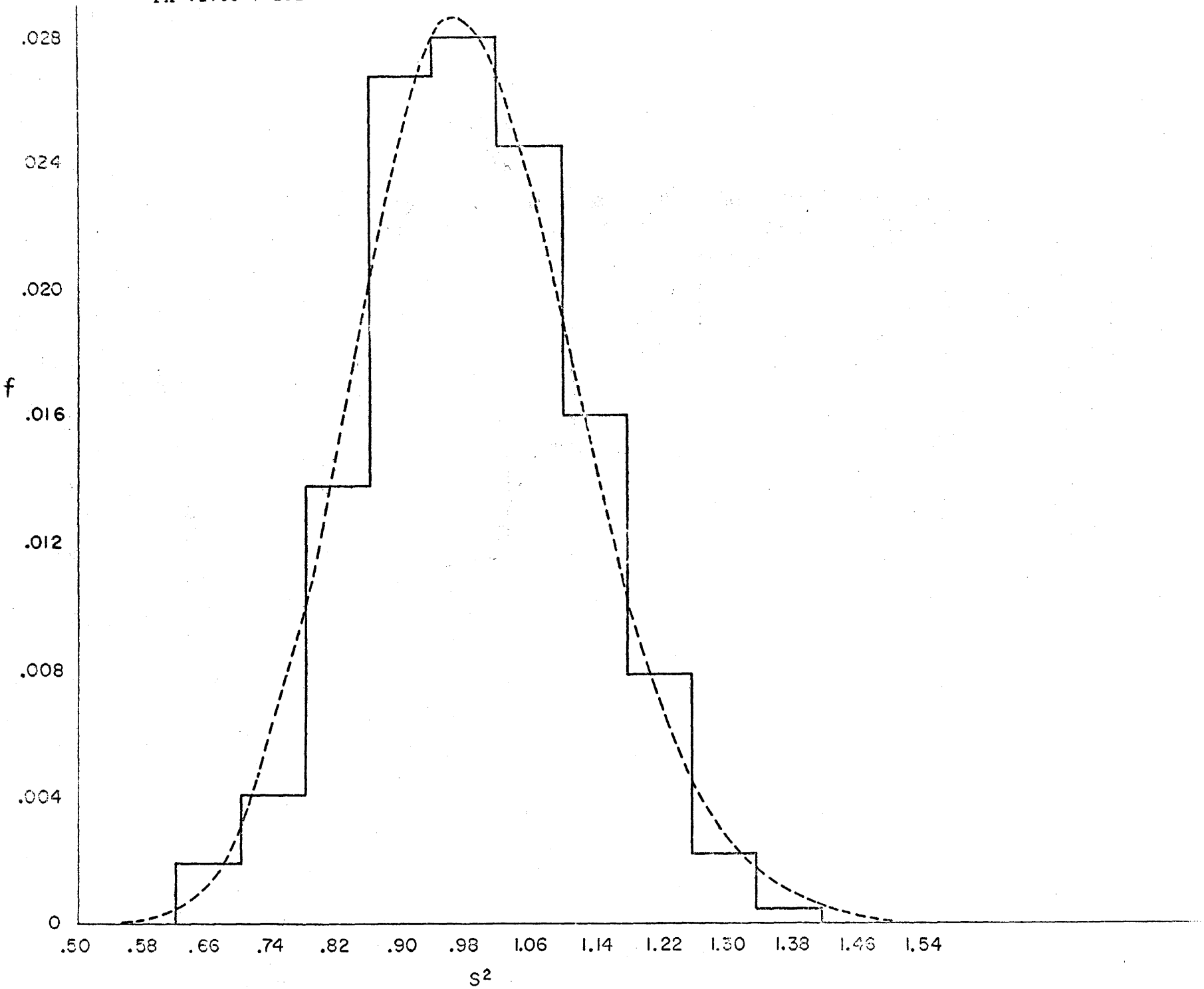


FIG. 2

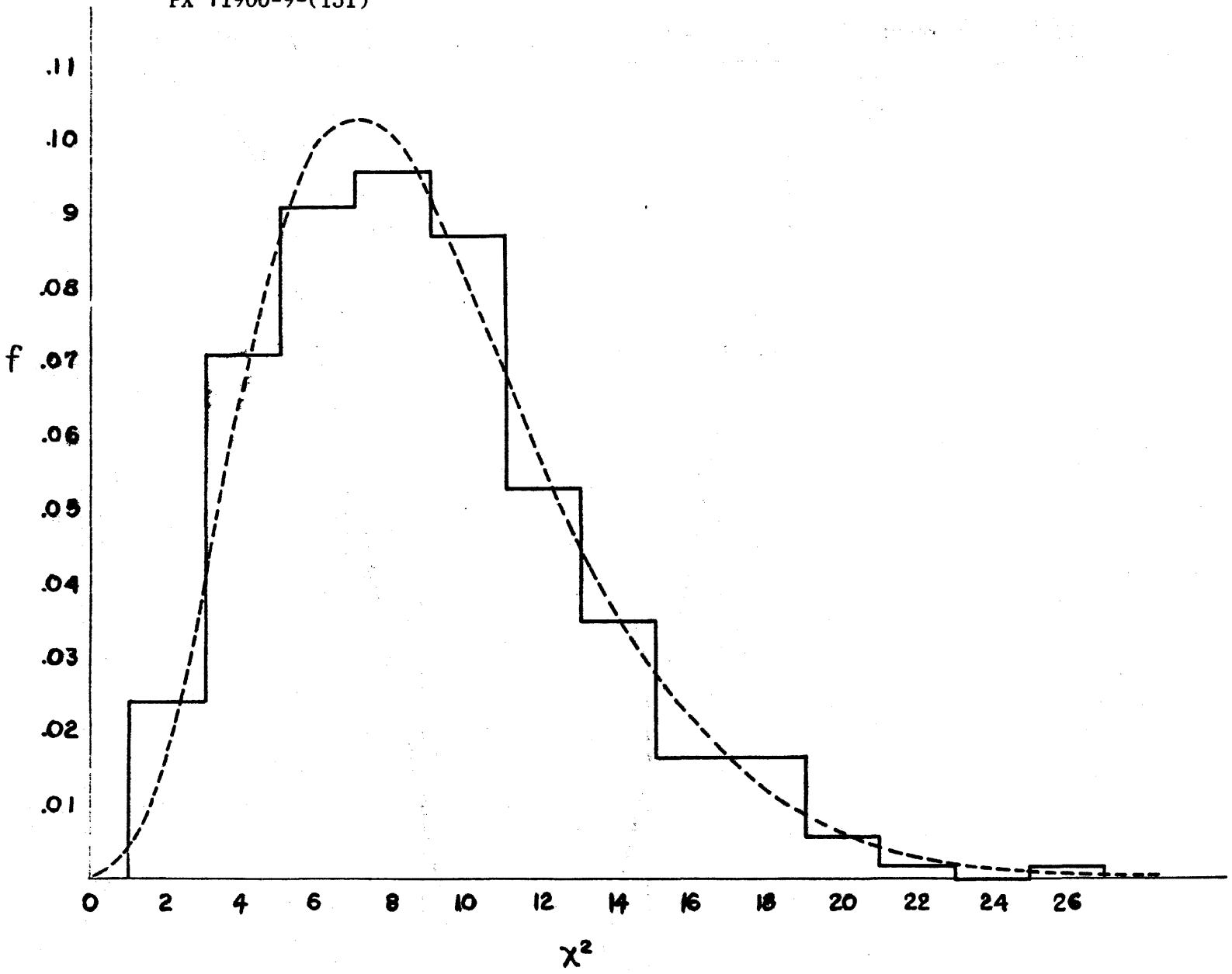


FIG. 3

PX 71900-9-(151)

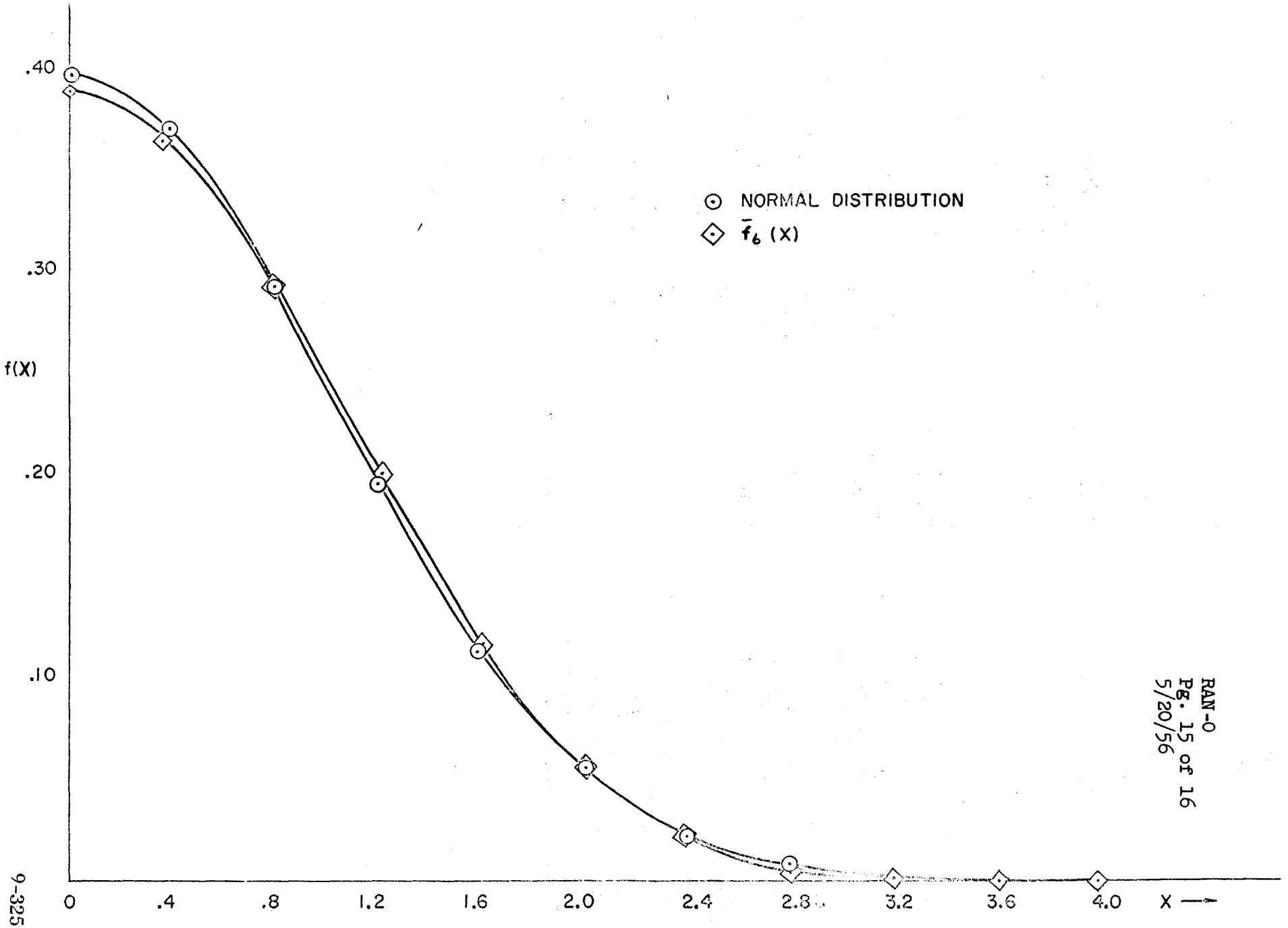


FIG. 4

RAN-0
Pg. 15 of 16
5/20/56

RW-151

9-325

D		00R00	01024		ROUTINE	02000	00	00000	00000
D		00T00	01037		TEMP STOR	02015	00	00000	00000
D		C1R00	00013		ROUTINE DRUM	63432	00	00000	00000
D		01T00	00003		TEMP DRUM	63447	00	00000	00000
D		01C80	00013			00015	00	00000	00000
D		00S00	00023			00027	00	00000	00000
1R00	MJ	00000	00000		EXIT	63432	45	00000	00000
1R01	TP	00T03	00T04		ENTRANCE 1	63433	11	02020	02021
1R02	TP	00T05	00S02		ENTRANCE 2	63434	11	02022	00031
1R03	TP	01C00	00S01			63435	11	00015	00030
1R04	XP	00T04	00T02		CALC AND	63436	71	02021	02017
1R05	TP	00T01	Q0000		STORE Y	63437	11	02016	10000
1R06	QT	A0000	00T04			63440	51	20000	02021
1R07	SS	00T00	00003			63441	34	02015	00104
1R08	AT	00S01	00S01		ADD Y TO S 1	63442	35	00030	00030
1R09	TJ	00S02	00R04			63443	41	00031	02004
1R10	MP	00S01	00T06		NORMALIZE	63444	71	00030	02023
1R11	TP	B0000	A0000			63445	11	30000	20000
1R12	MJ	00000	00R00			63446	45	00000	02000
1T00	20	00000	00000	B	MEAN	63447	20	00000	00000
1T01	37	77777	77777	B	MASK	63450	37	77777	77777
1T02	00	00000	00005	B	5 TO THE 5	63451	00	00000	00005
1T03	13	41437	54765	B	RESET YO	63452	13	41437	54765
1T04	13	41437	54765	B	Y	63453	13	41437	54765
1T05	00	00000	00005	B	N MINUS 1	63454	00	00000	00005
1T06	01	41421	31256		34 NORMAL CONS	63455	26	50126	76120

PX 71900-9-451)

THE RAMO-WOOLDRIDGE CORPORATION
Los Angeles 45, California

Column Heading Routine

Specifications

Identification Tag:	RPH-0
Type:	Service routine with program entrance
Entrance and Exit:	37 40020 40022 b to read a card
	37 40020 40023 b to punch a card

Coded by:	R. Beach	May, 1956
Approved by:	W. Bauer	May, 1956

Description

This routine will read in one card or punch out one card each time it is entered. It allows the programmer to include column headings in listed output when using either SNAP output or CPO-2. The routine stores cells 1660 b - 1777 b in the MD image, then bootstraps itself into ES for operation. After completion of the read or punch function, ES is restored and control returned to the program. A and Q are not altered by the routine.

Executing a read entry causes one card to be read and a complete alphanumeric card image to be stored. Executing a punch entry results in punching a card identical to the one read in with the exception of the twelve row of field three which is replaced with punches to control the 407 (See Operation). Only numeric information can be punched in field three and the information in this field will be printed by the 407 at the extreme left of the page, followed by 6 spaces.

A parameter word is used to specify the first of the 36 words in which the card image is to be located upon reading, or the first of the 36 words from which the card image to be punched.

The routine assumes that cards have been positioned in the Bull, but it does not alter card positioning when used. A blank card will be fed on the punch side when the read entry is executed.

Programming Instructions

1. To read a card. Enter the routine with 37 40020 40022 b followed by a parameter word of the form 00 00000 xxxxx where xxxxx is the address of the first cell in memory in which the card image is to be placed. Control is returned to the instruction following the parameter word.

2. To punch a card. Enter the routine with 37 40020 40023 b followed by a parameter word of the form 00 000YY xxxxx where xxxxx is the address of the first cell in memory at which the image to be punched is located, and YY is:

00	thru	19	for single spacing
20	thru	29	for double spacing
30	thru	49	for triple spacing
50	thru	89	for page ejection

The spacing takes place before printing the card. These page controls are identical to those used for SNAP with the exception that in this routine the combinations 00 thru 09 have the same effect as the combinations 10 thru 19.

The twelve row of field three of the card image will be replaced by a row containing the page controls and a SNAP identification punch (12 punch in column 75). It is advisable, therefore, to use only columns 1 thru 72 of the card for heading purposes. Only numeric information can be printed from field three. Since the card contains a SNAP identification punch, the word spacing in the first 72 columns will be the same as for SNAP; that is, four spaces are inserted after columns 12, 24, 36, 48, and 60. The numeric information in field 3 is printed at the extreme left of the page.

Restrictions

The card image may be located anywhere in memory except cells 1660 b - 1777 b (and also with the exception of cells reserved for the R-W library). It is recommended that card images be placed on the drum since the time involved in reading and punching using drum locations is very nearly the same as that involved when the card image is in ES.

Suggested Use

It is suggested that the input heading cards be placed at the front of the input deck, and that they be read in by the program before other computation is begun. Output heading cards can then be punched by the program at any time.

CONTROLLER

A CARD HANDLING SUBROUTINE IC 010

I. DESCRIPTION

This subroutine simplifies the reading and punching of cards especially in cases where blocks of cards with the same format are handled. It will be available shortly with the service routines on MT3.

This subroutine operates from the drum and handles all necessary block transfers. Up to 63 cards in each channel can be handled by one subroutine reference, reading and punching proceeding concurrently if desired. There are three entrances to this subroutine as described below.

II. CARD FORMATS

Each card format is described by a series of parameter words as described in IC 005, IC 006 and IC 007. For use with this subroutine, these words are preceded by one word, m, the number of fields on the card. These formats must be stored in MD. ($0 \leq m \leq 31$).

III. ENTRANCES

1. Prime:

37000000 70400

Next instruction

This subroutine is always one read card ahead of the main program. It retains the information from this card within itself. For this reason, when a new deck of cards is placed in the read hopper, the subroutine must be primed as well as the reproducer. Both are primed by this entrance to the subroutine.

2. READ

37 00000 70402

nn jjjjj vvvvv

Next instruction

This entrance cause nn cards of format jjjjj (first address for format) to be converted and stored in cells starting with vvvvv, (which may be in ES or MD).

3. PUNCH

37 00000 70401

nn jjjjj vvvvv

Next instruction

This entrance punches nn cards of format jjjjj using data starting in cell vvvvv, (which may be in ES or MD).

IV. OPERATING NOTES

1. If the "next instruction" after a punch card entrance is a read card entrance, the operations of reading and punching will begin simultaneously.

2. Whenever reading occurs without simultaneous punching, a blank card will appear in the output.

3. After approximately 270 words are read or punched, the reproducer will drop out of operation momentarily while block transfers are effected.

4. In case of trouble with the reproducer, start at 00000.

5. IC 007 is contained in this routine. It is located in cell 70656 and is modified to operate from location 00670.

V. SPECIFICATIONS

LOCATION: 70400 to 71257

TEMPORARIES: 74001 to 75777

This subroutine is not standard and is not to be modified. Its sum is constant.

CONTROLLER 6/4/56

```
70400          45 00000 70402
70401          45 00000 70402
70402          41 00000 00000     PICKUP ADDRESS
70403          75 31777 70405     SAVE
70404          11 00001 74001       ES
70405          11 20000 00002     STORE ADDRESS
70406          31 20000 00017     PICK
70407          15 20000 70411       UP
70410          75 30004 70412         CALLING
70411          11 30004 00422           SEQUENCE
70412          75 30660 00426     PLACE CONTROLLER
70413          11 70414 00426         IN ES
70414  00426  15 70410 70411     RESET DRUM SUM
70415  00427  16 00002 00635     SET EXIT
70416  00430  55 00422 10042     TEST
70417  00431  44 00433 00432       FIRST
70420  00432  44 00436 00621         ENTRANCE
70421  00433  11 00423 00425     NJV1 TO NJV 2
70422  00434  11 00640 00423     CLEAR NV1, 00641 TO J1
70423  00435  45 00000 00441
70424  00436  31 00424 00000     TEST SIMULTANEOUS
70425  00437  43 00642 00442         READ
70426  00440  11 00640 00425     CLEAR NV2, 00641 TO J2
70427  00441  23 00635 00643     ADJUST EXIT
70430  00442  15 00423 00466     SET FORMAT
70431  00443  15 00425 00464       PICKUPS
70432  00444  16 00423 00530     TEST
70433  00445  21 00530 00644       AND
70434  00446  42 00645 00450         SET
70435  00447  23 00530 00644           PUNCH
70436  00450  31 20000 00017             PICKUP
70437  00451  15 20000 00550               TO PIN
```

PX 71900-9-(153)

CONTROLLER 6/4/56

```
70440  00452  16 00425 00530     TEST
70441  00453  21 00530 00644       AND
70442  00454  42 00645 00456         SET READ
70443  00455  23 00530 00644           STORE
70444  00456  16 20000 00576             FROM BIN
70445  00457  31 00423 00052     N1 TO
70446  00460  11 20000 00426       M6
70447  00461  31 00425 00052     N2 TO
70450  00462  11 20000 00423       N6
70451  00463  75 30037 00465     PICKUP
70452  00464  11 00000 01740       READ FORMAT
70453  00465  75 30037 00467     PICKUP
70454  00466  11 00000 00430       PUNCH FORMAT
70455  00467  55 01740 20000     N1 LESS
70456  00470  36 00430 20000        M1
70457  00471  46 00472 00473     LARGER
70460  00472  11 00430 10000       TO Q
70461  00473  31 00646 00000     272 DIVIDED BY Q
70462  00474  73 10000 00422       TO N2
70463  00475  11 10000 00425          AND N2
70464  00476  11 00423 20000     COMPUTE
70465  00477  73 00422 00423       N3
70466  00500  47 00503 00501     TEST N4 EQUAL ZERO
70467  00501  23 00423 00641     ADJUST
70470  00502  11 00422 20000       N3 AND N 4
70471  00503  11 20000 00424     STORE N4
70472  00504  11 00423 20000     TEST
70473  00505  47 00507 00506       N3
70474  00506  11 00424 00422     N4 TO N2
70475  00507  11 00426 20000     COMPUTE
70476  00510  73 00425 00426       M3
70477  00511  47 00514 00512     TEST M4 EQUAL ZERO
```

PX 71900-9-(153)

CONTROLLER 6/4/56

```
70500  00512  23 00426 00641     ADJUST M3
70501  00513  11 00425 20000       AND M4
70502  00514  11 20000 00427     STORE M4
70503  00515  71 01740 00422     LOWER READ
70504  00516  11 20000 10000       STORE BY
70505  00517  23 00576 10000         N1 TIMES N2
70506  00520  11 00423 00467     N3 TO R
70507  00521  11 00426 00470     M3 TO S
70510  00522  21 00467 00641     N3-1 TO R
70511  00523  21 00470 00641     M3-1 TO S
70512  00524  11 00641 00471     1 TO P
70513  00525  11 00641 00472     1 TO Q
70514  00526  75 30037 00532     SHIFT READ
70515  00527  11 01740 00473       FORMAT
70516  00530  00 00000
70517  00531  00 00000 00 *
70520  00532  45 00000 00533     SWITCH A 00557
70521  00533  23 00472 00641     Q-1 TO Q
70522  00534  47 00560 00535     TEST
70523  00535  23 00470 00641     S-1 TO S
70524  00536  47 00537 00544     TEST
70525  00537  46 00540 00545     TEST
70526  00540  23 00611 00647     SUPPRESS PUNCHING
70527  00541  16 00650 00604     SET SWITCH E
70530  00542  16 00651 00532     SET SWITCH A
70531  00543  45 00000 00544     SWITCH B 00626
70532  00544  11 00427 00425     M4 TO M2
70533  00545  11 00425 00472     M2 TO Q
70534  00546  16 00652 00612     RESET PUNCH BIN
70535  00547  75 30420 00560     PICKUP TO
70536  00550  11 00000 00002       PUNCH BIN
70537  00551  16 00664 01012     PATCH
```

PX 71900-9-(153)

CONTROLLER 6/4/56

```
70540  00552  45 00000 00707      IN IC 007
70541  00553  16 00662 01012        CARD
70542  00554  45 00000 00704          SUBROUTINE
70543  00555  00 00000 00662
70544  00556  77 43000 10100    PARAMETER
70545  00557  16 00652 00612    RESET PUNCH BIN
70546  00560  45 00000 00561    SWITCH B 00605
70547  00561  23 00471 00641    P-1 TO P
70550  00562  43 00641 00564    TEST
70551  00563  47 00610 00571    TEST
70552  00564  11 00467 20000    TEST
70553  00565  47 00610 00566      R
70554  00566  11 00653 10000    SET TO PICK
70555  00567  53 00653 00611      PUNCH CARD
70556  00570  45 00000 00610        ON LAST READ
70557  00571  37 00571 00577    SWITCH D 00572
70560  00572  71 00473 00422    SET TO
70561  00573  32 00654 00017      STORE
70562  00574  15 20000 00575      N1 N2 WORDS
70563  00575  75 70000 00577    STORE FROM
70564  00576  11 01360 30000      READ BIN
70565  00577  23 00467 00641    R-1 TO R
70566  00600  47 00601 00605    TEST
70567  00601  46 00602 00606    TEST
70570  00602  23 00611 00656    SUPPRESS READING
70571  00603  16 00650 00343    SET SWITCH B
70572  00604  37 00560 00605    SWITCH E 00626, SET SWITCH C
70573  00605  11 00424 00422    N4 TO N2
70574  00606  16 00655 00611    RESET READ BIN
70575  00607  11 00422 00471    N2 TO P
70576  00610  37 00670 00670    ENTER IC 007
70577  00611  33 00474 01360      CARD
```

PX 71900-9-(153)

CONTROLLER 6/4/56

```
70600  00612  00 00431 00002        SUBROUTINE
70601  00613  21 00611 00473    RAISE BIN
70602  00614  21 00612 00430      ADDRESSES
70603  00615  21 00576 00473    RAISE
70604  00616  31 00430 00017      STORAGE
70605  00617  35 00550 00550        ADDRESSES
70606  00620  45 00000 00532
70607  00621  23 00635 00657    ADJUST EXIT
70610  00622  17 00000 00660    PICK CARDS
70611  00623  37 00670 00670    PICK CARDS
70612  00624  31 00556 00002      AND READ
70613  00625  00 00000 05 *         TO BCD BIN
70614  00626  23 20000 20000    SUM
70615  00627  75 20011 00631      BCD
70616  00630  32 01212 00000        BIN
70617  00631  13 20000 01223    ADJUST DRUM SUM
70620  00632  75 30012 00634    BCD BIN
70621  00633  11 01212 71200      TO DRUM
70622  00634  21 00635 00661    ADJUST EXIT
70623  00635  75 31777 00003    RESTORE ES
70624  00636  11 74001 00001      AND EXIT
70625  00637  00 00000    *     C
70626  00640  00 00641 00000      O
70627  00641  00 00000 00 01       N
70630  00642  37 00000 70402        S
70631  00643  00 00000 00 02         T
70632  00644  00 00000 74000          A
70633  00645  00 00000 76000           N
70634  00646  00 00000 00420            T
70635  00647  22 00000 00000             S
70636  00650  00 00000 00626
70637  00651  00 00000 00557
```

PX 71900-9-(153)

CONTROLLER 6/4/56

```
70640  00652  00 00000     2
70641  00653  20 00000 00000
70642  00654  00 00000 30000
70643  00655  00 00000 01360
70644  00656  11 00000 00000
70645  00657  00 00000 00 03
70646  00660  00 00000 00114
70647  00661  00 00000 00 04
70650  00662  75 10011 01013      CLEAR
70651  00663  11 00667 01212        BCD BIN
70652  00664  00 00000 00723
70653  00665  00 00000 0* *
70654  00666  00 00000 00 *
70655  00667  00 00000 00 0
70656  00670  71 01166 00611      START OF
70657  00671  15 20000 00724        IC 007 CARD
70660  00672  16 20000 00762          SUBROUTINE
70661  00673  16 01014 00000
70662  00674  55 20000 00003
70663  00675  31 01163 00003
70664  00676  52 01202 20000
70665  00677  32 20000 00001
70666  00700  44 00701 00701
70667  00701  44 01023 00702
70670  00702  16 01151 01010
70671  00703  44 00551 00553      JUMP TO PATCH ☐ CHANGE
70672  00704  37 01050 01011
70673  00705  41 00777 01042
70674  00706  45 01166 00611
70675  00707  37 01050 00714
70676  00710  76 00000 01266
70677  00711  76 10000 01226
```

PX 71900-9-(153)

CONTROLLER 6/4/56

```
70700  00712  76 10000 01252
70701  00713  45 00000 00721
70702  00714  32 01166 00000
70703  00715  37 00713 01011
70704  00716  41 01017 00766
70705  00717  16 00771 01017
70706  00720  37 00713 00766
70707  00721  15 01153 01143
70710  00722  37 01012 00706
70711  00723  37 00765 00724
70712  00724  55 00511 00000
70713  00725  44 00726 00727
70714  00726  16 00555 00765    ◘ CHANGE
70715  00727  55 10000 00013
70716  00730  51 01203 20000
70717  00731  16 20000 00753
70720  00732  55 10000 00006
70721  00733  51 01203 01123
70722  00734  55 10000 00006
70723  00735  51 01203 01117
70724  00736  55 10000 00006
70725  00737  51 01203 00777
70726  00740  32 20000 00016
70727  00741  35 01154 00742
70730  00742  00 00000 00 01
70731  00743  37 00743 00744
70732  00744  41 01123 01137
70733  00745  37 01150 00746
70734  00746  41 01117 01140
70735  00747  37 01144 00751
70736  00750  37 01150 00751
70737  00751  41 00777 01140
```

CONTROLLER 6/4/56

70740	00752	54	00742	10107
70741	00753	31	01123	00000
70742	00754	32	10000	00000
70743	00755	73	00742	01116
70744	00756	37	01144	01140
70745	00757	51	01161	20000
70746	00760	47	00761	00762
70747	00761	13	01116	01116
70750	00762	11	01116	01427
70751	00763	21	00724	01204
70752	00764	21	00762	01166
70753	00765	45	00000	01013
70754	00766	11	01153	00777
70755	00767	55	01236	00000
70756	00770	11	01202	01236
70757	00771	31	01165	00011
70760	00772	32	20000	00003
70761	00773	44	00774	00775
70762	00774	33	01017	00000
70763	00775	46	00776	00772
70764	00776	31	20000	00000
70765	00777	35	01223	01223
70766	01000	21	00777	01205
70767	01001	41	01236	00771
70770	01002	37	01002	01003
70771	01003	55	01252	00000
70772	01004	37	01002	00770
70773	01005	55	01266	00034
70774	01006	37	01002	00771
70775	01007	16	01014	00000
70776	01010	45	00000	01042
70777	01011	17	00000	00000

PX 71900-9-(153)

CONTROLLER 6/4/56

```
71000  01012  45 00000 00723      □ CHANGE
71001  01013  36 20000 00777
71002  01014  27 20000 00670
71003  01015  35 01166 01017
71004  01016  35 01165 00706
71005  01017  00 00000 00011
71006  01020  11 01170 01017
71007  01021  75 10044 01007      CLEAR IMAGE
71010  01022  11 00667 01223          ONLY □ CHANGE
71011  01023  32 01163 00000
71012  01024  37 01010 00703
71013  01025  15 20000 01033
71014  01026  32 20000 00016
71015  01027  15 20000 01067
71016  01030  11 01162 01267
71017  01031  11 01155 01270
71020  01032  37 01045 01033
71021  01033  55 00446 00000
71022  01034  44 01035 01051
71023  01035  37 01045 01051
71024  01036  16 01051 00777
71025  01037  75 30003 01041
71026  01040  16 00710 01045
71027  01041  37 01010 01044
71030  01042  75 20003 01044
71031  01043  23 01045 01166
71032  01044  16 01014 00000
71033  01045  77 00000 01253
71034  01046  77 10000 01223
71035  01047  77 10000 01237
71036  01050  45 00000 00710
71037  01051  55 10000 00013
```

PX 71900-9-(153)

CONTROLLER 6/4/56

71040	01052	51	01203	00000
71041	01053	33	20000	00000
71042	01054	37	01157	01070
71043	01055	27	00743	03732
71044	01056	44	01110	01057
71045	01057	16	01121	01136
71046	01060	37	01131	01061
71047	01061	41	01123	01124
71050	01062	31	01117	00017
71051	01063	35	01160	01064
71052	01064	37	77777	77530
71053	01065	31	01204	00023
71054	01066	73	00742	10000
71055	01067	12	00120	01123
71056	01070	31	01123	00043
71057	01071	32	10000	00000
71060	01072	73	01064	20000
71061	01073	35	20000	01064
71062	01074	37	01131	01075
71063	01075	41	01117	01102
71064	01076	16	01152	01131
71065	01077	41	00777	01113
71066	01100	41	00777	01132
71067	01101	15	01067	01104
71070	01102	21	01033	01204
71071	01103	21	01067	01204
71072	01104	16	01045	01131
71073	01105	33	01204	00000
71074	01106	55	00117	00000
71075	01107	44	01102	01124
71076	01110	27	01136	01060
71077	01111	43	01117	01121

PX 71900-9-(153)

CONTROLLER 6/4/56

71100	01112	45	00000	01124
71101	01113	31	01211	00000
71102	01114	35	01270	01116
71103	01115	35	01164	01117
71104	01116	27	01244	01267
71105	01117	00	00000	00 0
71106	01120	33	01210	00000
71107	01121	37	01136	01122
71110	01122	35	01270	01123
71111	01123	27	01255	01267
71112	01124	55	01267	00043
71113	01125	44	01126	01131
71114	01126	21	01270	01207
71115	01127	42	01156	01131
71116	01130	55	01267	00010
71117	01131	45	00000	01036
71120	01132	31	01064	00002
71121	01133	32	01064	00001
71122	01134	11	20000	01064
71123	01135	34	20000	00063
71124	01136	47	01121	01127
71125	01137	16	00743	01144
71126	01140	41	01017	01143
71127	01141	21	01143	01204
71130	01142	16	01130	01017
71131	01143	55	01212	00004
71132	01144	37	01144	01145
71133	01145	31	01123	00002
71134	01146	32	01123	00001
71135	01147	52	01206	01123
71136	01150	45	00000	00746
71137	01151	00	00000	01050

PX 71900-9-(153)

CONTROLLER 6/4/56

71140	01152	00	00000	01100
71141	01153	35	01212	01212
71142	01154	11	01166	00742
71143	01155	27	01225	01267
71144	01156	27	01253	01267
71145	01157	31	01123	00043
71146	01160	11	01167	01064
71147	01161	00	00000	00014
71150	01162	40	00000	00000
71151	01163	00	00000	00*02
71152	01164	00	00003	00000
71153	01165	54	00000	00000
71154	01166	00	00000	00*01
71155	01167	00	00000	00*01
71156	01170	00	00000	00*12
71157	01171	00	00000	00144
71160	01172	00	00000	01750
71161	01173	00	00000	23420
71162	01174	00	00003	03240
71163	01175	00	00036	41100
71164	01176	00	00461	13200
71165	01177	00	05753	60400
71166	01200	00	73465	45000
71167	01201	11	24027	62000
71170	01202	00	00000	00*03
71171	01203	00	00000	00077
71172	01204	00	00001	00000
71173	01205	00	00001	00001
71174	01206	00	00000	00017
71175	01207	00	00014	00000
71176	01210	00	00002	00000
71177	01211	00	00003	00000

PX 71900-9-(153)

CONTROLLER 6/4/56

```
71200  01212  00 00000 0* *       BINARY
71201  01213  00 00000 00 0
71202  01214  00 00000 00 *
71203  01215  00 00000 00 *         CODED
71204  01216  00 00000 00 0
71205  01217  00 00000 00 0
71206  01220  00 00000
71207  01221  00 00000 0  *
71210  01222  00 00000 00 *            DECIMAL BIN
71211  01223  00 00000 0  *       SUM ADJUSTER
```

ANALYSIS
PREPARED BY L. W. Barton
CHECKED BY
REVISED BY

CONVAIR
A DIVISION OF GENERAL DYNAMICS CORPORATION
SAN DIEGO

UNPACKED FLOATING POINT CARD READ

This routine reads up to five decimal floating point numbers from cards and stores them consecutively in unpacked form in either ES or MD.

By option the exponents may be ignored and the mantissas treated and stored consecutively as integers.

Drum address 72436 - 72751 inc.

Driver 72400 - 72435 inc.

Number of instructions 260 octal

Number of constants 34 octal

Number of temporaries 46 octal

ES address 01000 - 01361 inc.

COMMAND SEQUENCE:

The subroutine is coded to start in cell 01000 and is entered by the read sequence,

37 01001 01002

AB UUUUU VVVVV

in case of failure P A K 01000 will jump to last 37 read command for repeat.

It is also with driver to use the sequence

37 72400 72401

AB UUUUU VVVVV

which will store ES in 74001 - 75777 inc., and restore after use. With this driver sequence in case of failure P A K 72430 will restore ES and come to a 56 stop - release will then repeat the last read command.

No prime commands are included. 17 00000 72433 will prime one read card.

A 4, 5, or 7, do not print sum

any other number, print sum

B 4, 5, or 7, treat mantissas as consecutive integers

any other number, normal floating point.

UUUUU Address for storage of the first data word.

VVVVV Number of floating point numbers to be read or with option number of mantissas stored as integers.

CARD FORM COLUMNS:

1 punch in 12 row flag for exit 1-5 otherwise not used by routine.

6, 21, 36, 51, 66 decimal point, punched in card not used by routine.

7-16, 22-31, 37-46, 52-61, 67-76 ten decimal digit mantissa

17, 32, 47, 62, 77 sign of mantissa

A punch in any row except 0 or 12 will read as negative.

18-19, 33-34, 48-49, 63-64, 78-79

Exponent (power of ten) range -99 to 99

20 , 35, 50 , 65, 80 sign of exponent

A punch in any row except zero or 12 will read as negative.

Rounding at all operations occurs but only nine decimal digit accuracy may be assumed.

If the mantissa is zero the exponent will be stored unchanged in octal form.

A synthetic sum may be printed for each series of cards characteristic for t at series.

A 12 punch in row 1 will override the normal count and cause exit with the end of that card, In that case cell 01342 will be negative (with driver ES restoration will read over 01342)

The subroutine must be operated in ES but will store numbers addressed to ES or MD as read in.

UNPACKED FLOATING POINT CARD READ

```
7 400   00742   56 00000 30000     EXIT
7 401   00743   75 31777 72403     STORE
7 402   00744   11 00001 74001       ES
72403   00745   75 30352 00747     ROUTINE
72404   00746   11 72400 00742      TO ES
72405   00747   11 00742 20000     SET
72406   00750   36 01264 72432      REPEAT
72407   00751   35 01305 20000     SET
72410   00752   16 20000 00770      EXIT
72411   00753   55 00742 20025     TEST FOR
72412   00754   42 00775 00756      ES
72413   00755   45 00000 00757       ADDRESSES
72414   00756   32 00776 00000     ADDRESS MODIFICATION
72415   00757   55 20000 00017
72416   00760   37 00760 00761
72417   00761   16 20000 00763     SET ACQUISITION
72420   00762   11 00777 01024     ERASE INSTRUCTION
72421   00763   71 01264 30000     ACQUIRE CONTROL WORD
72422   00764   55 20000 00006
72423   00765   37 00760 00754     X
72424   00766   55 20000 00017
72425   00767   37 01001 01004     TP SIBRPITOME
72426   00770   75 31777 30000     RESTORE E S
72427   00771   11 74001 00001      AND EXIT
72430   00772   75 31777 72432     RESTORE E S
72431   00773   11 74001 00001      AND
72432   00774   30 00000 00000       REPEAT
72433   00775   02 00000 00104
72434   00776   74 00000 00000
72435   00777   11 00000 00000
72436   01000   30 00000 00000     REPEAT
72437   01001   45 00000 30000     EXIT
```

PX 71900-9-(154)

```
72440  01002  16 01001 01003
72441  01003  71 01264 30000     ACQUIRE CONTROL WORD
72442  01004  13 20000 01352     PRINT SUM FLAG
72443  01005  11 01263 01314     CLEAR
72444  01006  11 01263 01315       CELLS
72445  01007  16 20000 01314     NUMBER OF WORDS   N
72446  01010  55 20000 00003
72447  01011  13 20000 01351     READ AS INTEGER FLAG
72450  01012  31 01255 00000     SET FINAL
72451  01013  35 01277 01227      TRANSFER
72452  01014  11 01305 01356     SET TRANSFER
72453  01015  11 01265 01357       STEPS
72454  01016  44 01017 01022
72455  01017  23 01227 01313     MODIFICATINNS
72456  01020  11 01264 01356      FOR INTEGER
72457  01021  23 01357 01261       STORAGE
72460  01022  55 10000 00021     SET DATA
72461  01023  16 10000 01230      STORAGE
72462  01024  21 01001 01264     SET EXIT
72463  01025  36 01305 01000     SET REPEAT
72464  01026  23 01314 01264     N-1
72465  01027  46 01001 01030     EXIT IF N   0
72466  01030  41 01352 01032     SUM TEST
72467  01031  61 00000 01303     CARRIAGE RETURN
72470  01032  17 00000 01300     READ AND PICK CARC
72471  01033  11 01260 01350     LINE DIGIT   9
72472  01034  16 01256 01224     SET TEMPORARY STORAGE
72473  01035  75 10011 01037     CLEAR
72474  01036  11 01263 01316     MATRIX
72475  01037  76 00000 01361
72476  01040  76 10000 10000
72477  01041  76 10000 01360
```

PX 71900-9-(154)

```
72500  01042  37 01042 01043
72501  01043  54 01361 00034
72502  01044  11 01257 01054     SET 1ST STORAGE
72503  01045  11 01304 01327     SET INDEX   3
72504  01046  31 01306 00024     2 EXP 35
72505  01047  32 01263 00004     SHIFT 4
72506  01050  44 01051 01052     TEST BIT   0
72507  01051  32 01350 00000     ADD LINE DIGIT
72510  01052  46 01053 01047     TEST DIGITS   9
72511  01053  31 20000 00000     CLEAR A LEFT
72512  01054  30 00000 00000     STORE MATRIX WORD
72513  01055  21 01054 01307     STEP STORAGE
72514  01056  41 01327 01046     4 TIMES
72515  01057  37 01057 01060
72516  01060  11 01360 10000
72517  01061  37 01057 01045
72520  01062  11 01361 10000
72521  01063  37 01057 01046
72522  01064  37 01064 01065
72523  01065  23 01350 01264     REDUCE LINE DIGIT
72524  01066  46 01067 01037     TEST FOR 11 ROW
72525  01067  11 01264 01350
72526  01070  37 01064 01037     11 ROW
72527  01071  37 01042 01037     12 ROW, DUMMY
72530  01072  44 01073 01075     TEST EXIT FLAG
72531  01073  11 01302 01314     OVERRIDE INDEX
72532  01074  11 01255 01350     SET FLAG
72533  01075  31 01314 00000     N-1 REMAINDER
72534  01076  42 01261 01100     TEST LAST CARD
72535  01077  17 00000 01300     READ AND PICK CARD
72536  01100  11 01302 01332     SET INDEX FOR WORD CHANGE
72537  01101  15 01257 01123     SET FOR 1ST EXTRACTION
```

PX 71900-9-(154)

```
72540  01102  55 01316 00024     SHIFT FOR IDENT NUMBER
72541  01103  11 01302 01347     SET FOR 5 DATA WORDS
72542  01104  31 01304 00000     TEST FOR
72543  01105  42 01314 01119       LESS THAN
72544  01106  11 01314 01347       5 WORDS
72545  01107  31 01314 00017     N-1
72546  01110  35 01306 10000     TEST FOR
72547  01111  41 01351 01113       INTEGER
72550  01112  55 10000 00001         OPTION
72551  01113  31 01255 00000     SET SMALLER
72552  01114  35 10000 01227       STORAGE
72553  01115  37 01124 01120     SPACE FOR PERIOD
72554  01116  11 01260 01331     SET TALLY   9
72555  01117  11 01263 01327     CLEAR
72556  01120  41 01332 01123     TEST TO
72557  01121  21 01123 01306      CHANGE MATRIX
72560  01122  11 01262 01332       WORD
72561  01123  55 30000 00004     PSSITION DIGIT
72562  01124  37 01124 01125
72563  01125  31 01327 00002     N X 10
72564  01126  32 01327 00001      ADD
72565  01127  52 01301 01327       DIGIT
72566  01130  41 01331 01120     TEST N COMPLETE
72567  01131  11 01327 01334     SET EXPONENT FOR N   3
72570  01132  37 01124 01120      ACQUIRE
72571  01133  51 01301 20000      SZ¤PN
72572  01134  11 01265 01330       FLAG
72573  01135  47 01136 01140     TEST SIGN
72574  01136  13 01327 01334     SET EXPONENT FOR N   0
72575  01137  13 01265 01330     -FLAG
72576  01140  37 01140 01141
72577  01141  11 01327 01354     MANTISSA X 10 EXP 10
```

PX 71900-9-(154)

```
72600  01142  11 01330 01355    MANTISSA FLAG
72601  01143  11 01264 01331    SET TALLY
72602  01144  37 01140 01117    EXPONENT INFORMATION
72603  01145  41 01351 01217    TEST INTEGER OPTION
72604  01146  11 01263 01333    CLEAR
72605  01147  31 01354 00000    TEST N
72606  01150  47 01151 01223     FOR   0
72607  01151  23 01327 01330    EXPONENT
72610  01152  12 20000 20000     ADJUSTMENT
72611  01153  73 01265 01353    TENS DIGIT
72612  01154  31 20000 00017    UNITS DIGIT
72613  01155  35 01254 01215    UNITS POWER
72614  01156  11 20000 01251     OF 10
72615  01157  23 01334 01265    ADJUST EXPONENT
72616  01160  11 01276 01327    10 EXP 10
72617  01161  11 01310 01334    ADJUST AXPONENT
72620  01162  46 01214 01250    SIGN OF EXPONENT
72621  01163  23 01334 01311    ADJUST EXPONENT
72622  01164  31 01354 00000    N
72623  01165  73 01327 01333    N ADJUSTMENT
72624  01166  31 20000 00043     FOR NEGATIVE
72625  01167  73 01327 10000       EXPONENT
72626  01170  55 10000 00001    DETERMINE
72627  01171  32 01263 00001     LAST
72630  01172  42 01327 01174       BIT
72631  01173  27 10000 01264         OF N
72632  01174  31 10000 00044    ASSEMBLE
72633  01175  32 01333 00044     M
72634  01176  11 01263 01331    CLEAR
72635  01177  74 20000 01331    SCALE FACTOR N
72636  01200  11 20000 01354    STORE N
72637  01201  46 01202 01207    TEST FOR ROUNDING
```

PX 71900-9-(154)

```
72640  01202  21 01354 01264     ROUND
72641  01203  43 20000 01207     ADJUST FOR
72642  01204  32 01263 00107      OVERFLOW
72643  01205  11 20000 01354       IF
72644  01206  21 01334 01264        NECESSARY
72645  01207  31 01331 00000     ADJUST
72646  01210  42 01303 01212
72647  01211  23 01334 01312      SCALE
72650  01212  21 01334 01331       FACTOR
72651  01213  37 01213 01214
72652  01214  41 01353 01163     TEST TENS DIGIT
72653  01215  30 00000 00000     POWER FOR UNITS DIGIT
72654  01216  37 01213 01163     ADJUST FOR UNITS DIGIT
72655  01217  11 01354 01333     N
72656  01220  41 01355 01222     SIGN OF N
72657  01221  13 01333 01333     NEGATIVE
72660  01222  21 01315 01333     SUM
72661  01223  75 30002 01225     STORE
72662  01224  11 01333 30000      TEMPORARY
72663  01225  21 01224 01356     STEP STORAGE
72664  01226  41 01347 01115     TEST END OF CARD
72665  01227  30 00000 00000     FINAL
72666  01230  11 01335 30000      STORAGE
72667  01231  21 01230 01357     STEP
72670  01232  23 01314 01261     TEST FOR
72671  01233  46 01234 01033      END
72672  01234  41 01352 01001     TEST PRINT SUM
72673  01235  31 01263 00000     CLEAR
72674  01236  75 00006 01240     SUM
72675  01237  32 01315 00014
72676  01240  31 20000 00030
72677  01241  32 01264 00004
```

```
72700  01242  61 00000 20000
72701  01243  34 20000 00000
72702  01244  47 01241 01001     PRINTING
72703  01245  71 01354 01327    ADJUSTMENT
72704  01246  37 01213 01176     OF N
72705  01247  37 01247 01250      AND
72706  01250  41 01353 01245       EXPONENT
72707  01251  30 00000 00000        FOR
72710  01252  37 01247 01245         POSITIVE
72711  01253  45 00000 01217           SCALE FACTOR
72712  01254  11 01264 01327    CONSTANTS
72713  01255  75 30000 01231
72714  01256  00 00000 01335
72715  01257  35 01316 01316
72716  01260  00 00000 00*11
72717  01261  00 00000 00005
72720  01262  00 00000 00010
72721  01263  00 00000 00*0
72722  01264  00 00000 00001
72723  01265  00 00000 00012
72724  01266  00 00000 00144
72725  01267  00 00000 01750
72726  01270  00 00000 23420
72727  01271  00 00003 03240
72730  01272  00 00036 41100
72731  01273  00 00461 13200
72732  01274  00 05753 60400
72733  01275  00 73465 45000
72734  01276  11 24027 62000
72735  01277  00 00012 00000
72736  01300  00 00000 00105
72737  01301  00 00000 00017
```

PX 71900-9-(154)

72740	01302	00	00000	00004
72741	01303	00	00000	00045
72742	01304	00	00000	00003
72743	01305	00	00000	00002
72744	01306	00	00001	00000
72745	01307	00	00001	00001
72746	01310	00	00000	00043
72747	01311	00	00000	00044
72750	01312	00	00000	00110
72751	01313	00	00005	00000

ARC SIN AND ARC COS, FIXED POINT CF015

01000: Alarm exit

01001: Exit

01002: Entry

Initial State: $(A) = N \cdot 2^{34}$

where $\sin\theta = N$

and $\cos\phi = N$

$|N| \leq 1$

Final STate: $\theta \cdot 2^{33} \longrightarrow (A)$

$\phi \cdot 2^{33} \longrightarrow (Q)$

FORMULA USED: Series expansion about $\frac{\pi}{4}$

$$\theta = \frac{\pi}{4} + N^* + \sum_{m=1}^{i} \frac{U_m}{2m+1}$$

$$U_m = \frac{2m-1}{2m} N^{*2} U_{m-1}$$

$$N^* = \frac{1}{\sqrt{2}}\left(N - \sqrt{1-N^2}\right)$$

Convergence is assumed when $U_m < 2^{-34}$

Maximum time; 70.5 milliseconds

Drum address: 73464 through 73551 $(54)_{10}$ $(66)_8$

Number of commands for Assembly Modification: $(50)_{10}$ $(62)_8$

Uses 60 cells including constants and temporaries 01066 - 01073 inclusive.

Special tests: If N = 0 θ is set = 0

ϕ is set = $\frac{\pi}{2}$

If $|N| > 1$ exit to Alaram Print Out

PX 71900-9-(155)

73464	01000	37	76000	76002	ALARM
73465	01001	45	00000	30000	EXIT
73466	01002	11	01063	10000	$\pi/2 \cdot 2^{33} \longrightarrow (Q)$
73467	01003	47	01004	01001	$N = 0$?
73470	01004	16	01042	01055	SET FOR POSITIVE RESULT
73471	01005	46	01006	01007	N NEGATIVE ?
73472	01006	16	01024	01055	SET FOR NEGATIVE RESULT
73473	01007	12	20000	01067	STORE $\lvert N \rvert$
73474	01010	31	01064	00041	$1.2^{34} \longrightarrow (A)$
73475	01011	42	01067	01000	$\lvert N \rvert > 1$? $\longrightarrow$ ALARM
73476	01012	33	01064	00103	$-1.2^{68} \longrightarrow (A)$
73477	01013	72	01067	10000	$(-1+N^2) \cdot 2^{68} \longrightarrow (A)$
73500	01014	47	01015	01030	$(A) = 0$?
73501	01015	13	20000	01066	STORE $(1-N^2) \cdot 2^{68} = \overline{N} \cdot 2^{68}$
73502	01016	34	20000	00044	
73503	01017	13	20000	01073	
73504	01020	31	01043	00000	$X_0 \longrightarrow (A)$
73505	01021	11	20000	01070	STORE X_i
73506	01022	31	01073	00044	$\overline{N} \cdot 2^{68} \longrightarrow (A)$
73507	01023	32	01066	00000	
73510	01024	73	01070	20000	$\overline{N}/X_0 \longrightarrow (A)$
73511	01025	32	01070	00107	$\frac{1}{2}(N/X_0 + X_0) = X_1$
73512	01026	11	20000	20000	
73513	01027	42	01070	01021	$X_0 > X_1$?
73514	01030	34	01067	00000	$(\sqrt{1-N^2} - N) \cdot 2^{34}$
73515	01031	71	20000	01062	$\sqrt{2}/2 (N - \sqrt{1-N^2}) \cdot 2^{68} = N^* \cdot 2^{68}$
73516	01032	54	20000	00046	$N^* \cdot 2^{34} \longrightarrow (R)$

PX 71900-9-(155)

73517	01033	71	20000	10000	$N^{*2} \cdot 2^{68} \longrightarrow (A)$, $N^{*} \cdot 2^{34} \longrightarrow (Q)$
73520	01034	54	20000	00046	STORE $N^{*2} \cdot 2^{34}$
73521	01035	11	20000	01066	
73522	01036	13	01065	01067	$D_0 = -1$
73523	01037	11	10000	20000	$N^{*} \cdot 2^{34} \longrightarrow (A)$
73524	01040	11	20000	01073	STORE $N^{*} \cdot 2^{34}$
73525	01041	11	01063	01072	$\pi/4 \cdot 2^{34}$
73526	01042	35	01072	01072	
73527	01043	21	01067	01064	$D + 2 \longrightarrow D = 2m-1$
73530	01044	35	01065	01070	$D + 1 \longrightarrow P = 2m$
73531	01045	35	01065	01071	$P + 1 \longrightarrow Q = 2m+1$
73532	01046	71	01066	01073	$N^{*2} \cdot U_{m-1} \cdot 2^{68} \longrightarrow (A)$
73533	01047	54	20000	00046	$N^{*2} \cdot U_{m-1} \cdot 2^{34}$
73534	01050	71	20000	01067	$(2m-1)\, N^{*2} \cdot U_{m-1} \cdot 2^{34}$
73535	01051	73	01070	20000	$2m-1/2m\ N^{*2} \cdot U_{m-1} = U_m$
73536	01052	11	20000	01073	STORE U_{m-1}
73537	01053	73	01071	20000	$U_m/2m+$
73540	01054	47	01042	01055	$U_m/2m+1 \neq 0$ REPEAT LOOP
73541	01055	13	01072	01072	SWITCH
73542	01056	54	01072	00107	$-\theta \longrightarrow (A)$
73543	01057	35	01063	10000	$\pi/2 - \theta \longrightarrow (Q)$
73544	01060	13	01072	20000	$\theta \longrightarrow (A)$
73545	01061	45	00000	01001	JUMP TO EXIT
73546	01062	64	53730	31462	$-\sqrt{2}/2 \cdot 2^{34}$
73547	01063	14	44176	65210	$\pi/4 \cdot 2^{34}$
73550	01064	00	00000	00002	
73551	01065	00	00000	00001	

PX 71900-9-(155)

LEASE SQUARES POLYNOMIAL APPROXIMATION

PART I. INTRODUCTION

This routine is intended to enable the programmer to approximate a function y of λ independent variables $x_1, x_2, \cdots x_\lambda$ by a polynomial $P=\sum_{i_1+i_2+\cdots+i_\lambda}^{d} a_{i_1 i_2 \cdots i_\lambda} x_1^{i_1} x_2^{i_2} \cdots x_\lambda^{i_\lambda}$ of degree d if y is known for n+1 sets of the independent variables. The code is written for $\lambda = 4$ and $d = 2$. By treating powers of variables as independent variables and making a few modifications in the code it is possible to handle approximations for various combinations of λ and d (including for example, $\lambda = 1$ and $d = 14$).

PART II. ANALYSIS

($\lambda = 4$, $d = 2$, and $n+1 = 45$). Let the known values of y be denoted by ${}^{\mu}y$ and those of the independent variables by ${}^{\mu}x_1, {}^{\mu}x_2, {}^{\mu}x_3, {}^{\mu}x_4$. It is desired to determine the a_{ijkl} so as to minimize

$$(1)\quad \sum_{\mu=0}^{n} \left[{}^{\mu}y - \sum_{i+j+k+l=0}^{2} a_{ijkl} \left(x_1^i x_2^j x_3^k x_4^l \right)_\mu \right]^2 \equiv \sum_{\mu=0}^{n} d_\mu^2$$

Differentiating (1) with respect to the unknowns a_{ijkl} and setting the results to zero yields

$$(2)\quad \begin{cases} 2 \sum_{\mu} d_\mu \left[-\left(x_1^\circ\ x_2^\circ\ x_3^\circ\ x_4^\circ \right)_\mu \right] = 0 \\ 2 \sum_{\mu} d_\mu \left[-\left(x_1^\prime\ x_2^\circ\ x_3^\circ\ x_4^\circ \right)_\mu \right] = 0 \\ \cdots\cdots\cdots\cdots \\ 2 \sum_{\mu} d_\mu \left[-\left(x_1\ x_2\ x_3\ x_4 \right)_\mu \right] = 0, \end{cases}$$

which, if simplified becomes

$$(3)\quad \begin{cases} a_{0000} \sum_{\mu} \left(x_1^\circ x_2^\circ x_3^\circ x_4^\circ \right)_\mu + a_{1000} \sum_{\mu} \left(x_1^\prime x_2^\circ x_3^\circ x_4^\circ \right)_\mu + \cdots + a_{0002} \sum_{\mu} \left(x_1^\circ x_2^\circ x_3^\circ x_4^{\prime\prime} \right)_\mu = \sum_{\mu} {}^{\mu}y \cdot \left(x_1^\circ x_2^\circ x_3^\circ x_4^\circ \right)_\mu \\ a_{0000} \sum_{\mu} \left(x_1^\prime x_2^\circ x_3^\circ x_4^\circ \right)_\mu + a_{1000} \sum_{\mu} \left(x_1^{\prime\prime} x_2^\circ x_3^\circ x_4^\circ \right)_\mu + \cdots + a_{0002} \sum_{\mu} \left(x_1^\prime x_2^\circ x_3^\circ x_4^{\prime\prime} \right)_\mu = \sum_{\mu} {}^{\mu}y \cdot \left(x_1^\prime x_2^\circ x_3^\circ x_4^\circ \right)_\mu \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ a_{0000} \sum_{\mu} \left(x_1^\circ x_2^\circ x_3^\circ x_4^{\prime\prime} \right)_\mu + a_{1000} \sum_{\mu} \left(x_1^\prime x_2^\circ x_3^\circ x_4^{\prime\prime} \right)_\mu + \cdots + a_{0002} \sum_{\mu} \left(x_1^\circ x_2^\circ x_3^\circ x_4^{\prime\prime} \right)_\mu = \sum_{\mu} {}^{\mu}y \cdot \left(x_1^\circ x_2^\circ x_3^\circ x_4^{\prime\prime} \right)_\mu \end{cases}$$

or (in matrix-vector form)

$$(4)\quad \begin{pmatrix} \sum_\mu (x_1^0 x_1^0 x_3^0 x_4^0)_\mu & \sum_\mu (x_1^1 x_1^0 x_3^0 x_4^0)_\mu & \cdots & \sum_\mu (x_1^0 x_2^0 x_3^0 x_4^2)_\mu \\ \sum_\mu (x_1^1 x_2^0 x_3^0 x_4^0)_\mu & \sum_\mu (x_1^2 x_2^0 x_3^0 x_4^0)_\mu & \cdots & \sum_\mu (x_1^1 x_2^0 x_3^0 x_4^2)_\mu \\ \cdots & \cdots & \cdots & \cdots \\ \sum_\mu (x_1^0 x_2^0 x_3^0 x_4^2)_\mu & \sum_\mu (x_1^1 x_2^0 x_3^0 x_4^2)_\mu & \cdots & \sum_\mu (x_1^0 x_2^0 x_3^0 x_4^4)_\mu \end{pmatrix} \begin{pmatrix} a_{0000} \\ a_{1000} \\ \cdot \\ a_{0002} \end{pmatrix} = \begin{pmatrix} \sum_\mu y_\mu (x_1^0 x_2^0 x_3^0 x_4^0)_\mu \\ \sum_\mu y_\mu (x_1^1 x_2^0 x_3^0 x_4^0)_\mu \\ \cdot \\ \sum_\mu y_\mu (x_1^0 x_2^0 x_3^0 x_4^2)_\mu \end{pmatrix}$$

The determinant of the system (4) is non-singular and symmetric. Thus, the problem is completed by solving the non-homogeneous linear system (4) by Cramer's Method. For the general case the number ν of equations in (4) is given by (5) $\nu = \binom{\lambda-1}{0} + \binom{\lambda}{1} + \cdots + \binom{\lambda+d-1}{d}$. For the present case $\nu = 15$. The Crout routine is used to solve (4). For convenience denote the so-called given matrix in (4) by M, and M without summation signs by m.

PART II. CODING

The code is developed by combining a least squares code (called 2X4) with existing sub-routines. 2X4 includes a "residual" calculation (evaluation of (1) for calculated a_{ijkl}). The following list gives the parts of the tape in the order in which they appear on the tape:

IC 003 FLOATING POINT CARD INPUT (followed by extra seventh level hole).

MD CLEAR 40000-47777 (followed by extra seventh level hole).

2 X 4 FIT.

CM 005 CROUT

CA 001 TWO REGISTER FLOATING POINT ARITHMETIC

IC 004 FLOATING POINT CARD OUTPUT

The register of the CROUT routine, 77251, indicating the number of equations is set to $(17)_8 = (15)_{10}$ and the exit is modified to jump to the residual check

PX 71900-9-(156)

of the 2 X 4 FIT. The n index register, 50775, of the 2 X 4 routine is set to $(54)_8 = (44)_{10}$. There is an MS:1 at 444 upon completion of the generation of m. There is an MS:2 at 466 at which point m has been generated and stored and the routine is ready to enter the CROUT routine.

PART III INPUT - OUTPUT

Input and output is by cards in floating decimal format (five numbers to the card). The input cards contain $'x_1, 'x_2, 'x_3, 'x_4, 'y$ followed by $^2x_1, ^2x_2, ^2x_3, ^2x_4, ^2y$ and so on, with a header card punched to start the loading at 51003. The output cards contain five sets of answers a_{ijkl} in the order 0000, 1000, 0100, 0010, 0001, 2000, 1100, 1010, 1001, 0200, 0110,...,0002, each set followed by their differences (see description of CROUT routine) followed by a card containing the residual for the last set of answers.

PART IV. OPERATION

1. Load service routines.
2. Load input cards into read hopper.
3. PT load 2 X 4 to first extra 7th level hole.
4. (PAK) = 300, start 1103.
5. PT load 2 X 4 to second extra 7th level hole.
6. MDS to clear 40000-47777.
7. PT load remaining portion of 2 X 4.
8. (PAK) = 50376, start 1103.

The total time for the case $\lambda = 4$, $d = 2$, $n = 44$ is approximately twelve minutes.

LEAST SQUARES POLY APPRX

```
51001          20 00000 00000  1 INTO
51002          00 00000 00001    STORAGE

50376  00376   75 30400 00400  TRANSFER ROUTINE
50377  00377   11 50400 00400    TO ES
50400  00400   75 30002 00402  1  TO ES
50401  00401   11 51001 01001
50402  00402   75 30010 00404  X_i TO ES
50403  00403   11 51003 01003
50404  00404   75 30002 00406  X_i TO (27-30)
50405  00405   11 01003 00027
50406  00406   75 30002 00410  X_i TO (25-26)
50407  00407   11 01003 00025
50410  00410   37 00201 00203  MULTIPLY
50411  00411   75 30002 00413
50412  00412   11 00031 01013  ELEMENT OF m1 TO ES
50413  00413   21 00407 00770  ADVANCE (407_u)
50414  00414   21 00412 00041  ADVANCE (412_v)
50415  00415   41 00763 00406   4-i  TIMES ?
50416  00416   21 00405 00770  ADVANCE (405_u)
50417  00417   15 00405 00407  (407_v) = (405_u)
50420  00420   23 00415 00073  DIMINISH (415_u) BY 1
50421  00421   41 00771 00404  4 TIMES ?
50422  00422   75 30002 00424  Y TO ES
50423  00423   11 51013 01037
50424  00424   75 30040 00426  ELEMENTS OF 1ST ROW TO RESIDUAL
50425  00425   11 01001 52271    CHECK STORAGE AREA
50426  00426   11 00765 00764  SET LOOP INDEX
50427  00427   75 30002 00431  ELEMENT OF ROW 1 AS
50430  00430   11 01003 00027    MULTIPLIER
```

PX 71900-9-(156)

LEAST SQUARES POLY APPRX

```
50431  00431  75 30002 00433  ELEMENTS OF ROW 1 AS
50432  00432  11 01001 00025    MULTIPLICANDS
50433  00433  37 00201 00203  MULTIPLY
50434  00434  75 30002 00436  ELEMENTS OF ROWS
50435  00435  11 00031 01041    2-15 TO ES
50436  00436  21 00432 00770  ADVANCE (432u)
50437  00437  21 00435 00041  ADVANCE (435v)
50440  00440  41 00764 00431  ONE ROW OF m COMPLETED ?
50441  00441  21 00430 00770  ADVANCE (430u)
50442  00442  15 00766 00432  (432u) = 1001
50443  00443  41 00767 00426  14 ROWS OF m COMPLETED ?
50444  00444  56 10000 00445  TO m ACCUMULATION
50445  00445  75 30002 00447  ACCUMULATE
50446  00446  11 01001 00025    ELEMENTS
50447  00447  75 30002 00451      OF  m
50450  00450  11 40001 00027        IN AN ES AREA
50451  00451  37 00201 00202          PREPARATORY TO
50452  00452  75 30002 00454            TRANSFER TO
50453  00453  11 00031 01001              GIVEN
50454  00454  21 00446 00770                MATRIX
50455  00455  21 00450 00770                  AREA OF
50456  00456  21 00453 00041                    CROUT
50457  00457  41 00773 00445                      ROUTINE
50460  00460  75 30740 00462  [M]TO DRUM
50461  00461  11 01001 40001
50462  00462  21 50403 00772  ADVANCE (50403u)
50463  00463  21 50423 00772  ADVANCE (50423u)
50464  00464  21 50425 00774  ADVANCE (50425v)
50465  00465  41 00775 50376  ALL POINTS USED ?
50466  00466  56 20000 77330  TO CROUT
50760  00760  00 00000 00000    ?
50761  00761  00 00000 00001    X
```

PX 71900-9-(156)

LEAST SQUARES POLY APPRX

```
50762  00762  00 00000 00002      4
50763  00763  00 00000 00003       R.
50764  00764  00 00000 00017        T.
50765  00765  00 00000 00017  C
50766  00766  00 01001 00000   O
50767  00767  00 00000 00015    N
50770  00770  00 00002 00000     S
50771  00771  00 00000 00003      T
50772  00772  00 00012 00000       A
50773  00773  00 00000 00357        N
50774  00774  00 00000 00040         T
50775  00775  00 00000 00054          S
55776         75 10004 56000  CLEAR RESIDUAL
55777         11 00040 00664   STORAGE REGISTERS
56000         75 30060 00610  TRANSFER RESIDUAL CHECK
56001         11 56002 00602   ROUTINE TO ES
56002  00602  00 00000 00054  RESIDUAL
56003  00603  00 00040 00000   CHECK
56004  00604  00 76500 00000    CALCU -
56005  00605  00 00000 00016     LATION
56006  00606  00 00000 00016      CONSTANTS
56007  00607  11 00605 00606  SET INDEX
56010  00610  75 30002 00612  POWERS TO
56011  00611  11 52271 00027   (27-30)
56012  00612  75 30002 00614  COEFFICIENTS TO
56013  00613  11 76500 00025   (25-26)
56014  00614  37 01001 01003  MULTIPLY
56015  00615  75 30002 00617  TERMS TO
56016  00616  11 00031 00027   (27-30)
56017  00617  75 30002 00621  PARTIAL SUM
56020  00620  11 00664 00025    OF POLY. TO (25-26)
56021  00621  37 01001 01002  ACCUMULATE
```

PX-71900-9-(156)

LEAST SQUARES POLY APPRX

```
56022  00622  75 30002 00624  STORE PARTIAL SUM
56023  00623  11 00031 00664    OF POLYNOMIAL
56024  00624  21 00611 00661  ADVANCE (611u)
56025  00625  21 00613 00661  ADVANCE (613u)
56026  00626  41 00606 00610  POLY. EVALUATED
56027  00627  13 00664 00027  DIFFERENCE
56030  00630  11 00665 00030   BETWEEN
56031  00631  75 30002 00633    APPROXIMATION
56032  00632  11 52327 00025     AND
56033  00633  37 01001 01002      FUNCTION
56034  00634  75 30002 00636  CALCULATION
56035  00635  11 00031 00027   OF
56036  00636  75 30002 00640    THE
56037  00637  11 00031 00025     SQUARE
56040  00640  37 01001 01003      OF
56041  00641  75 30002 00643       ONE
56042  00642  11 00031 00027        DIFFERENCE
56043  00643  75 30002 00645  PARTIAL
56044  00644  11 00666 00025   SUM
56045  00645  37 01001 01002    OF
56046  00646  75 30002 00650     THE
56047  00647  11 00031 00666      RESIDUAL
56050  00650  21 00632 00603  ADVANCE (632u)
56051  00651  15 00604 00613  (613u) = 76500
56052  00652  21 00611 00661  ADVANCE (611u)
56053  00653  75 10002 00655
56054  00654  11 00040 00664  CLEAR PARTIAL SUM STOR. REG.
56055  00655  41 00602 00607  ALL POINTS USED ?
56056  00656  37 70440 70441  PUNCH
56057  00657  00 00666 00001    RESIDUAL
56060  00660  57 00000 00000  FINAL STOP
56061  00661  00 00002 00000  INCREMENT
```

PX 71900-9-(156)

FIXED POINT CHARACTRON OUTPUT ROUTINE

This routine prints numbers on the charactron in decimal form, given the binary scale factor.

Drum address 73032 - 73322 inc.

Driver 72770 - 73031 inc.

Number of instructions: 213 octal

Constants: 56 octal

Temporaries: 7 octal

ES address: 01000 - 01277

COMMAND SEQUENCE

The subroutine is coded to start in cell 01000 and is entered by the sequence:

37 01000 01000

NN UUUUU VVVVV

If the driver is to be used, it is entered by the sequence:

37 72770 72771

NN UUUUU VVVVV

The driver will store ES in 74001 - 75777 inc. and restore it after use. In case of failure with driver sequence, starting with PAR equal to 73022 will stop print, advance the film, restore ES, and come to a 56 stop. Restarting will then repeat the interrupted printing instruction.

NN is the maximum number of rows from the top which will be printed. If it is zero or over 32 it will be set equal to 32.

The number of columns is 4

UUUUU is the address of the first data word

VVVVV is the address of the first parameter word

PARAMETER WORDS

The parameter words consist of pairs of octal digits from left to right corresponding to consecutive data words. If the value of the octal digit pair is not greater than octal 43 it is used as a binary scale factor for the corresponding data word. If the value is octal 77 it serves as a flag to end the parameter words and exit. If it is greater than octal 43 and not equal to octal 77, E will precede that data word which will have its binary scale factor set equal to zero. Parameter words are not disturbed in use.

PAGE FORMAT

A title and page number are placed on each page. The title is any number of symbols up to 48. Two octal digits represent each symbol as defined by this table. Any other pair will cause a blank space.

		Second Octal Digit					
		(0)	(1)	(2)	(3)	(4)	(5)
First	(0)	0	1	2	3	4	5
	(1)	6	7	8	9	A	B
Octal	(2)	C	D	E	F	G	H
	(3)	.	J	K	L	M	N
	(4)	-	P	Q	R	S	T
Digit	(5)	U	V	W	X	Y	Z

Cells 01251 - 01260 (or with driver 73303 - 73312) inclusive are used to store the title. The page number is stored in cell 01214 (and by the driver also in cell 73246.) It normally is read in as zero and advanced by one each time.

A maximum of 128 numbers per page is possible in four columns of 32 numbers each. The columns have an even left edge preceeded by negative

signs when required. Leading zeros are suppressed, the decimal point is printed and the maximum size may reach 11 digits. Integers over 9,999,999,999 are replaced by that value. Numbers with fractional parts are rounded. Any column is completed as far as specified before the next is started.

PAGE AND FILM ADVANCE

Cell 01213 is used for grid position in printing and cell 01214 is used for page number (the driver also uses 73246 and 73247). Both of these are read in zero and are automatically modified.

The film is advanced at any time the subroutine is entered with cell 01214, page number, equal to zero, is before the first printing. Film is also advanced at the end of each page. At the end of each page 01213 is set equal zero. This will cause a new page to be started when further printing is asked for.

Both 01214 and 01213 can be set equal zero to cause film advance and/or a new page. Film advance also will occur if the instruction 17 00000 01241 (or with driver 17 00000 73273) is used. This should be executed twice at the end of a problem to develop and fix the last two frames.

Unless cell 01213 is cleared or the end of a page is reached a succeeding 37 command sequence will start printing the next word on the same page.

With the driver, addition of 72032 to any above BS address will give the corresponding MD address.

TIMING

Between succeeding pages .4 seconds occurs as the film advances (and with the developing camera 2.seconds for development). If there is less than 2½ seconds computing time per page, a delay may hold up the computer. (These times are approximate and subject to change).

CHARACTRON FIXED POINT WRITE

```
72770  00736  67 00001 30000
7 771  00737  75 31777 72773      ENTRANCE MD DRIVER
7 772  00740  11 00001 74001       STORE ES
7 773  00741  75 30333 00743     ROUTINE
7 774  00742  11 72770 00736      TO ES
7 775  00743  16 00736 00766     SET
72776  00744  21 00766 01216      EXIT
72777  00745  55 00736 20025     TEST
73000  00746  44 00751 00750      FOR
73001  00747  45 00000 00751       ES
73002  00750  32 00776 00000        ADDRESS
73003  00751  55 20000 00017          ADJUSTMENT
73004  00752  37 00752 00753
73005  00753  16 20000 00757     SET ACQUISITION
73006  00754  11 00736 73026     SET REPEAT
73007  00755  11 00777 01017     ERASE INST
73010  00756  16 00775 01173     SET EXIT
73011  00757  71 01216 30000     CONTROL WORD
73012  00760  55 20000 00006     ADJUST
73013  00761  37 00752 00746      ES
73014  00762  37 00752 00746         ADDRESSES
73015  00763  45 00000 01001     SO ROUTINE
73016  00764  11 01213 73245     RESTORE GRID NUMBER, RETURN
73017  00765  11 01214 73246     RESTORE PAGE NUMBER
73020  00766  75 31777 30000     RESTORE
73021  00767  11 74001 00001       E.S.
73022  00770  17 00000 73273     ADVANCE FILM
73023  00771  75 31777 73025     RESTORE
73024  00772  11 74001 00001       ES
73025  00773  23 73026 73031
73026  00774  30 00000 00000     REPEAT
73027  00775  02 00000 00764
```

PX 71900-9-(157)

CHARACTRON FIXED POINT WRITE

```
73030  00776  74 00000 00000
73031  00777  11 00001 00001
73032  01000  71 01216 30000     ACQUIRE CONTROL WORD
73033  01001  15 20000 01055     ADDRESS OF 1ST DATA WORD
73034  01002  55 20000 00006
73035  01003  51 01231 20000     NUMBER OF ROWS
73036  01004  55 10000 00011
73037  01005  15 10000 01020     ADDRESS OF 1ST S.W.
73040  01006  47 01007 01010     TEST NUMBER ROWS   0
73041  01007  42 01270 01011     TEST NUMBER ROWS GREATER 32
73042  01010  31 01216 00005     SET    32
73043  01011  71 20000 01234      CHANGE TO
73044  01012  35 01216 01273       DISTANCE
73045  01013  31 01214 00000     PAGE NUMBERS
73046  01014  47 01016 01015     TEST    0
73047  01015  17 00000 01241     ADVANCE FILM
73050  01016  21 01000 01216     SET
73051  01017  16 20000 01172       EXIT
73052  01020  11 30000 01275     STORE 1ST PW
73053  01021  15 01020 01102     TRANSFER IDENTITY
73054  01022  11 01246 01274     SET FOR PW CHANGE
73055  01023  17 00000 01237     FREE RUN PRINT
73056  01024  31 01213 00000     TEST FOR
73057  01025  47 01046 01026       NEW PAGE
73060  01026  15 01211 01027     SET PICKUP FOR TITLE
73061  01027  11 30000 10000     PICKUP
73062  01030  11 01246 01271     SET TALLY
73063  01031  51 01261 20000     EXTRACT FROM Q
73064  01032  55 10000 00006     SHIFT CHARACTERS
73065  01033  32 01213 00000     ADD POSITION
73066  01034  77 10000 20000     PRINT
73067  01035  21 01213 01233     SHIFT RIGHT
```

CHARACTRON FIXED POINT WRITE

```
73070  01036  41 01271 01031    TEST TALLY
73071  01037  21 01027 01244    ADVANCE PICKUP
73072  01040  42 01210 01027    TEST END
73073  01041  21 01214 01216    ADVANCE PAGE NUMBER
73074  01042  11 20000 01272    TO INTEGER STORAGE
73075  01043  11 01245 01271    0 TO FRACTIONAL STORAGE
73076  01044  37 01134 01110    PRINT PAGE NUMBER
73077  01045  11 01234 01213    SET GRID POSITION
73100  01046  55 01275 00006    STORE SCALE FACTOR
73101  01047  51 01231 01277     TEST FOR
73102  01050  43 01231 01171       EXIT FLAG
73103  01051  42 01242 01055    TEST FOR OVERSIZE
73104  01052  31 01243 00000    E FOR SCALE FACTOR ERROR
73105  01053  11 01215 01277    SET SCALE FACTOR   0
73106  01054  37 01205 01201    PRINT E
73107  01055  11 30000 01271    ACQUIRE NUMBER
73110  01056  21 01055 01244    STEP
73111  01057  31 01207 00000    SET BINARY
73112  01060  36 01277 01103      POINT SHIFT
73113  01061  11 01271 20000
73114  01062  46 01063 01067    TEST FOR SIGN
73115  01063  13 01271 01271    NEGATIVE
73116  01064  31 01213 00000    GRID
73117  01065  32 01245 00000    NEGATIVE SIGN
73120  01066  77 10000 20000    PRINT
73121  01067  21 01213 01233    MOVE SIGHT
73122  01070  71 01277 01267    .2857 X BSF
73123  01071  32 01236 00052      .99
73124  01072  11 20000 01276    NUMBER FRACTIONAL DIGITS
73125  01073  31 20000 00017    SHIFT TO U
73126  01074  35 01212 01076    ROUNDING
73127  01075  31 01262 00000      FACTOR
```

PX 71900-9-(157)

CHARACTRON FIXED POINT WRITE

```
73130  01076  30 00000 00000        TO Q
73131  01077  41 01274 01103    TEST FOR PW CHANGE
73132  01100  11 01246 01274    RESET TALLY
73133  01101  21 01102 01244    ADVANCE PW
73134  01102  11 30000 01275    NEXT PW
73135  01103  30 00000 00000    CENTER BINARY POINT
73136  01104  32 10000 00000    ROUND
73137  01105  11 20000 01271    STORE FRACTION
73140  01106  34 20000 00044
73141  01107  11 20000 01272    STORE INTEGER
73142  01110  75 20013 01112    JUMP IF OVERSIZE
73143  01111  42 01216 01114    JUMP IF SIZE OK
73144  01112  11 01247 01272    SET FOR MAX. INTEGER
73145  01113  11 01215 10000
73146  01114  51 01231 01277    11-R
73147  01115  55 01277 10017    SET
73150  01116  31 01206 00000
73151  01117  36 10000 01124      DIVISION
73152  01120  31 01217 00000    10
73153  01121  36 01277 01277    R-1  NUMBER INTEGER DIGITS
73154  01122  31 01272 00044    REPLACE
73155  01123  32 01271 00000      NUMBER
73156  01124  30 00000 00000    DIVISOR IN Q
73157  01125  11 10000 01272
73160  01126  73 10000 10000    ANSWER X 2 EXP 35
73161  01127  55 10000 00001    ANSWER X 2 EXP 36
73162  01130  32 01215 00001    REMAINDER X 2
73163  01131  42 01272 01133    TEST WITH DIVISOR
73164  01132  27 10000 01216    ROUND
73165  01133  37 01205 01204    PRINT INTEGER
73166  01134  37 01134 01135    OPTIONAL EXIT
73167  01135  31 01250 00000    PERIOD
```

CHARACTRON FIXED POINT WRITE

```
73170  01136  32 01213 00000    GRID
73171  01137  77 10000 20000    PRINT PERIOD
73172  01140  21 01213 01233    MOVE RIGHT
73173  01141  11 01276 01277    CHANGE TO FRACTION TALLY
73174  01142  37 01205 01204    PRINT FRACTION
73175  01143  11 01213 10000    GRID
73176  01144  51 01264 20000    MASKED
73177  01145  35 01234 01213    ADVANCE VERTICAL
73200  01146  11 01263 10000    V MASK
73201  01147  51 01213 20000    VERTICAL ADJUSTMENT
73202  01150  42 01273 01046    JUMP IF OK
73203  01151  16 01234 01213    VERTICAL TO START
73204  01152  21 01213 01266    ADVANCE HORIZONTAL
73205  01153  15 01213 01277    HORIZONTAL
73206  01154  31 01277 00000      TO U OF A
73207  01155  42 01265 01046    JUMP IF OK
73210  01156  17 00000 01240    STOP PRINT
73211  01157  11 01215 01213    SET FOR NEW PAGE
73212  01160  17 00000 01241    ADVANCE FILM
73213  01161  15 01046 01166    TEST
73214  01162  31 01274 00000     FOR
73215  01163  47 01166 01164
73216  01164  15 01102 01166    END
73217  01165  21 01166 01244     OF
73220  01166  13 30000 10000
73221  01167  51 01261 20000    PAGE
73222  01170  47 01023 01172     EXIT
73223  01171  17 00000 01240    STOP PRINTING
73224  01172  45 00000 30000    EXIT
73225  01173  31 10000 00002      Q   X 10
73226  01174  32 10000 00001       TO A
73227  01175  11 20000 10000    REMAINDER TO A
```

PX 71900-9-(157)

```
73230  01176  34 20000 00102     OVERFLOW TO POSITION
73231  01177  42 01235 01201     TEST FOR
73232  01200  35 01232 20000       ADJUSTMENT
73233  01201  32 01213 00000     GRID
73234  01202  77 10000 20000     PRINT DIGIT
73235  01203  21 01213 01233     MOVE TO RIGHT
73236  01204  41 01277 01173     END OF WORD
73237  01205  45 00000 30000     SUB EXIT
73240  01206  55 01230 10001     FOR NUMBER ADJUSTMENT
73241  01207  31 01271 00044     FOR BINARY POINT SHIFT
73242  01210  11 01261 00000     FOR PAGE TITLE TEST
73243  01211  00 01251 00000
73244  01212  73 01216 10000     FOR ROUNDING FACTOR
73245  01213  00 00000 00*0     GRID LOCATION
73246  01214  00 00000 00*0     PAGE NUMBER
73247  01215  00 00000 00 *     ZERO
73250  01216  00 00000 00001     POWERS
73251  01217  00 00000 00012
73252  01220  00 00000 00144
73253  01221  00 00000 01750
73254  01222  00 00000 23420        OF
73255  01223  00 00003 03240
73256  01224  00 00036 41100
73257  01225  00 00461 13200
73260  01226  00 05753 60400
73261  01227  00 73465 45000
73262  01230  11 24027 62000           10
73263  01231  00 00000 00077     MASK
73264  01232  02 00000 00000
73265  01233  00 00022 00000     HORIZONTAL SPACE
73266  01234  00 00000 00036     VERTICAL SPACE
73267  01235  06 00000 00000     TEST
```

PX 71900-9-(157)

```
73270  01236  00 77270 24366    .99 X 2 EXP 30
73271  01237  00 00000 05000    PRINT FREE RUN
73272  01240  00 00000 04400    STOP PRINT
73273  01241  00 00000 06000    ADVANCE FILM
73274  01242  00 00000 00044    36
73275  01243  22 00000 00000    E
73276  01244  00 00001 00000
73277  01245  40 00000 00000    PRINT NEGATIVE SIGN
73300  01246  00 00000 00005
73301  01247  11 24027 61777    10 EXP 10 -1
73302  01250  30 00000 00010    PRINT PERIOD
73303  01251  44 70077 00770    CHARACTERS
73304  01252  07 70077 00770
73305  01253  07 70077 00770      FOR
73306  01254  35 00527 72522
73307  01255  14 43774 52501        PAGE
73310  01256  44 70077 00770
73311  01257  07 70077 00770          TITLE
73312  01260  07 70077 00770    MASK
73313  01261  77 00000 00000
73314  01262  37 77777 77777
73315  01263  00 00000 77777    MASK
73316  01264  00 07400 77777    MASK
73317  01265  00 02000 00000
73320  01266  00 00400 00000
73321  01267  00 22221 64247    .2857 X 2 EXP 30
73322  01270  00 00000 00041
```

PX 71900-9-(157)

REMINGTON RAND UNIVAC

St. Paul, Minnesota

15 April, 1956

FLICK, A DEMONSTRATION ROUTINE

Description

This routine enables one to display a message in Times-Square fashion across the monitor scope on the 1103 console, or on paper tape from the high speed punch. The routine is provided with program entries and exits for use as a subroutine. (HSS) are preserved in the region 76000-77777 while FLICK operates and are restored upon completion of the display.

Input

The routine will read a message from paper tape which has been punched in flex code directly from a Flexowriter; the routine will also display a message which has been previously stored on the drum starting at address 50000. In either case, the upper limit on the number of characters per message is 2047_{10}. A flex code-stop (octal 43) must terminate a message.

The flex-coded characters recognized by FLICK are A thru Z, ∅ thru 9, minus (-), equality (=), plus (+), period (.), and space. Space codes are automatically substituted for carriage return and tab codes. FLICK ignores all other flex codes, and paper tape leader.

In addition, a number of other characters are available for display. These characters cannot be punched on tape directly from a Flexowriter, and hence use of them requires that they be read into the computer independently of FLICK. The additional characters and their octal codes are as follows:

CHAR.	CODE	CHAR.	CODE	CHAR.	CODE	CHAR.	CODE
x	131	∈	121	∅(upper case)	100	8(upper case)	110
÷	112	√‾	122	1 "	101	9 "	111
:	113	•	123	2 "	102	<	132
?	114	(	124	3 "	103	>	133
!	115	)	125	4 "	104	≥	134
∫	116	[	126	5 "	105	≤	135
Σ	117	]	127	6 "	106	&	136
π	120	/(upper case)	130	7 "	107		

A bioctal tape of character codes for a desired message may be loaded into the computer, one character to a cell starting at address 50000. This form of message input must of course be effected independently of FLICK routine.

Output

Each character of the message is displayed in a 5x7 matrix on the scope, or on the paper tape. One character moves across the scope in approximately 3 seconds, and characters are punched out at the rate of 10 per second.

Storage

The FLICK routine is located in cells 47161 thru 47777. The coded message is stored in cells 50000 thru 50000+n, where n is the number of characters in the message. Cells 54000 thru 54000+n are used as temporary storage. $n \leq 2047_{10}$ (3777_8). The FLICK routine is available in symbolic coding (RECO) so that rearrangement of storage can be readily performed.

Any message remains undisturbed on the drum once it is read in; this enables one to redisplay a message as many times as is desired until a new message is read in.

Operating Instructions

READ MESSAGE FROM	TYPE OF DISPLAY	PAK SETTING*	PROGRAM ENTRY
Flex-coded paper tape	scope	47324	37 47312 47316
Drum (50000 thru 50000+n)	scope	47321	37 47312 47313
Flex-coded paper tape	punch	47173	37 47161 47165
Drum (50000 thru 50000+n)	punch	47170	37 47161 47162

*Note: If only one display is desired, MS1 select should be turned ON at this point; otherwise the current display will repeat indefinitely. Program entry to FLICK gives only one complete message display.

Alarms

1. If an illegal character code is encountered, the computer halts on a 56 00077 00163 command. STARTing will terminate the current message and begin display of those characters which up to the stop point have been found to be legal ones.

2. If a flex code-stop (octal 43) is omitted from the end of the message, FLICK will continue to read from tape or drum until an illegal code is found (see alarm 1.). When the code-stop is omitted from punched paper tape, the computer may hang up on an Eternal Read command; in this case Force Stop, set PAK = 00136 and START. The message is then automatically terminated and display begins.

A listing of the code in symbolic-relative (for assembly by RECO) and octal and/or a bioctal paper tape suitable for reading into the 1103 may be obtained upon request addressed to:

Leo B. Kennedy
Univac Scientific
Applications Group

CC - A Useful Instruction for Inverted Binary Numbers

Binary counters and converters will sometimes supply a quantity in the inverted binary or gray code. When converting from ordinary binary notation to gray code or vice versa, the CC - command may be used conveniently.

Conversion from ordinary to gray code can be done with these 2 instructions:

a	LA	x	A + 71	Shift ordinary numbers 1 right and leave in A
a + 1	CC	x	A	Add to each bit the following bit modulo 2

This takes care of numbers up to 36 bits long. Execution time is 228 μ sec. The reverse process is not quite so elegant, it uses repeated LT and CC.

a	SP	x	36-n	Scale up inverted numbers
a + 1	LTL	o	x	Clear cell x
a + 2	RPV	n	a + 4	Make a string of n shifted
a + 3	LTL	1	T	Words
a + 4	RPV	n	a + 6	Add to each bit all preceding
a + 5	CC	x	T	bits modulo 2.

The size of the number to be converted is n. The execution time is (270 + 56n) μ sec.

Both procedures can be easily adapted to put results in other cells.

Robert G. Tantzen,
Holloman Air Development Center

THE RAMO WOOLDRIDGE CORPORATION
Los Angeles 45, California

ARCTAN

Specifications

Identification Tag:	TNI-0
Type:	Subroutine
Assembly Routine Spec:	SUB 50277 07214
Storage:	58 instructions
	14 constants in program
	72 words of total program storage used
	4 words of temporary storage pool used, addresses 00027b through 00032b
Entrance and Exit:	RJ SUB00 SUB01
	The alarm exit is not used.
Drum Assignment:	64045b through 64154b
Machine Time:	3.7 ms average machine time
Mode of Operation:	Fixed point

Coded by:	P. Johnson	December 3, 1954
Code Checked by:	T. Tack	March 15, 1955
Machine Checked by:	M. Elmore	March 19, 1955
Approved by:	W. Dixon	April 29, 1955

PX 71900-9-(160)

Description

This routine computes F(X), an approximation to the arctangent of X, by using a polynomial approximation. It was programmed by Engineering Research Associates and was adopted for use at Ramo-Wooldridge.

Programming Instructions

Assume the routine is stored at SUB00, and that X is the number whose arctangent is desired.

1. Place $X \cdot 2^{33}$ in the accumulator
2. Execute RJ SUB00 SUB01

Control will be returned to the cell immediately following the return jump and $F(X) \cdot 2^{33}$ will be left in the accumulator. $-\frac{\pi}{2} \leq F(X) \leq \frac{\pi}{2}$

Mathematical Method and Error Analysis

The Rand Approximation

$$\text{Arctangent } X \doteq F(X) = \sum_{i=0}^{7} C_{2i-1} X^{2i-1}$$

is used. (See Rand Sheet 13.) The accuracy as stated by ERA is

$$\left|\text{Arctangent } X - F(X)\right| \leq 2^{-25}$$

Machine Checking

Sixteen values of the argument were tested in the range $-1225 \leq X \leq 1225$.

PX 71900-9-(160)

THE RAMO-WOOLDRIDGE CORPORATION
Los Angeles 45, California

Gill Method Subroutine

Specifications

Identification Tag:	NUI-3
Type:	Subroutine
Assembly Storage Spec:	SUB 49880 07414
Storage:	59 instructions, addresses OGM00 thru OGM40 1GM00 thru 1GM17 15 constants in program, addresses OGC00 thru OGC14 74 words total program storage, addresses OGM00 thru OGM40 1GM00 thru 1GM17 OGC00 thru OGC14 10 words temporary storage pool used, addresses 00027b (OGT00) thru 00040b (OGT09)
Drum Assignment:	Addresses 63230b thru 63341b
Program Entrances:	Addresses OGM02, OGM03, and OGM04
Program Exit:	Address OGM01
Machine Time:	(10.3 n + 1.9) ms per point average, where n equals the number of equations in the system
Mode of Operation:	Fixed point
Coded by:	J. Carlson R. Douthitt M. Elmore R. Summers
Code Checked by:	M. Elmore — June 8, 1955
Machine Checked by:	M. Elmore — July 7, 1955
Approved by:	W. Bauer — July 22, 1955

PX 71900-9-(161)

Description

The Gill Method Subroutine integrates a system of first order, differential equations using a step-by-step process. Using the values of the variables at a point and the coding for computing the derivative of each of the dependent variables at that point, the Gill Method Subroutine produces the coordinates for the next point of the solution each time it is entered.

A special entrance sets up the subroutine for a particular system of equations, thus allowing the subroutine to solve concurrently several different systems in the same program.

The independent variable is incremented within the subroutine itself.

Notation

The system of equations to be solved is

$$\frac{dy_i}{dx} = f_i (x, y_1, y_2, \ldots, y_n), (i = 1, 2, \ldots, n).$$

q_i are intermediate values of the calculation (zero initially)

Δx is the increment of the independent variable x

h is the binary scaling power of x (i.e. $x \cdot 2^h$ is in the computer)

h-1 is the binary scaling power of Δx

m_i is the binary scaling power of y_i

f is the common difference between the scaling power of y_i and the scaling power of $\frac{dy_i}{dx}$ for each i.

m_i - f is the binary scaling power of $\frac{dy_i}{dx}$

L = 73 + f - h

Programming and Operating Instructions

Assign the Gill Method Subroutine to some arbitrary region, say OGMOO.

In order to solve a given system, the following array of variables, derivatives, intermediate values, and parameters should be assigned a region, say OGNOO.

OGN00		L	
OGN01	00 OGN05 OGN06		
OGN02		n-1	
OGN03		Δx	scaled 2^{h-1}
OGN04		x	scaled 2^{h}
OGN05		$\frac{dy_1}{dx}$	scaled m_1 - f
OGN06		y_1	scaled m_1
OGN07		q_1	initially zero
OGN08		$\frac{dy_2}{dx}$	scaled m_2 - f
OGN09		y_2	scaled m_2
OGN10		q_2	initially zero

.
.
.

In addition, the coding for computing $\frac{dy_i}{dx}$ for all i, (i = 1, 2, . . . , n) should be assigned a region, say ODE00. This coding will use the values in region OGN00 to compute all $\frac{dy_i}{dx}$ as specified by the equations in the system and should place the results in the appropriate places in region OGN00. It should then exit to the Gill Method Subroutine with an MJ 00000 OGM04 (see below).

Assuming the Gill Method Subroutine is in region OGM00, the three entrances are OGM02, OGM03, and OGM04. The exit is OGM01.

PX 71900-9-(161)

The first entrance, OGM02, is used for setting up the Gill Method Subroutine only for the particular system to be solved. It is entered by an RJ command followed by a parameter word which specifies the location of the variables, and the location of the coding for calculating the derivatives:

RJ OGM01 OGM02
00 OGN00 ODE00

The second entrance, OGM03, is the entrance for producing a point of the solution. It is entered by an RJ command: RJ OGM01 OGM03. Entering using this command results in four passes through both the Gill Method Subroutine and the coding for computing the derivatives, and leaves in region OGN00 the new values of the variables, the derivatives at those values, and x advanced by Δx, ready for the next step.

The third entrance, OGM04, is the entrance from the coding for calculating the derivatives and is used on each of the four passes necessary for computing one point. As noted above, it is entered by an MJ command in the ODE00 region:

MJ 00000 OGM04

Mathematical Analysis

Theory. "A Process for the Step-by-Step Integration of Differential Equations in an Automatic Digital Computing Machine" by S. Gill, published in Cambridge Philosophical Society Proceedings, Vol. 47, Part I, January 1951, should be consulted for a detailed analysis of the process on which the subroutine is based.

Suppose we know the point $(X, Y_1, Y_2, \ldots, Y_n)$ on the curve defined by the system of equations

$$\frac{dy_1}{dx} = f_1 (x, y_1, y_2, \ldots, y_n)$$

$$\frac{dy_2}{dx} = f_2 (x, y_1, y_2, \ldots, y_n)$$

$$\vdots$$

$$\frac{dy_n}{dx} = f_n (x, y_1, y_2, \ldots, y_n)$$

The Gill Method is a process by which we can find the next point on the curve: i.e. the value of $y_1, y_2, \ldots, y_n$ for $x = X + h$.

The process can be better understood if the case where n=1 is first considered.

We have the point (X,Y) on the curve $\frac{dy}{dx} = f(x,y)$, and we want to find y at $X + h$; i.e. we want $k = \delta y$ such that $\left.\frac{dy}{dx}\right]_{X + h,\ Y + k} = f(X + h, Y + k)$.

We derive k by making four approximations and averaging them in a particular way.

First approximate the curve by a straight line through (X,Y) with the slope $\left.\frac{dy}{dx}\right]_{X,Y} = f(X,Y)$, and find a first approximation to k:

$$k_o = h \cdot f(X,Y)$$

Then we travel a fraction m of the way along this line to the point $(X + mh, Y + mk_o)$ and find $f(X + mh, Y + mk_o)$.

This gives us a new straight line through $(X + mh, Y + mk_o)$ with slope $f(X + mh, Y + mk_o)$, and we find

$$k_1 = h\, f(X + mh, Y + mk_o)$$

We now use k_o and k_1 to find a third point at which f is calculated: $(X + nh, Y + [n-r]\, k_o + rk_1)$.

$$k_2 = h\, f(X + nh, Y + [n-r]\, k_o + rk_1)$$

Similarly,

$$k_3 = h \cdot f\ (X + ph,\ Y + \left[p-s-t\right]\ k_o + sk_1 + tk_2$$

The weighted average of k_o, k_1, k_2, and k_3 is the desired $k = \delta y$:

$$\delta y = y\ (X + h) - y\ (X) = c_o k_o + c_1 k_1 + c_2 k_2 + c_3 k_3$$

where $c_o + c_1 + c_2 + c_3 = 1$.

For a system of equations, the same four steps given above are made for each equation and

$$\delta y_i = c_o k_{io} + c_1 k_{i1} + c_2 k_{i2} + c_3 k_{i3} \text{ where } c_o + c_1 + c_2 + c_3 = 1.$$

The above process is, for certain values of m, n, p, s, t, c_o, c_1, c_2, and c_3, the Runge-Kutta process. The Gill process was derived, with application to machine use in mind, by minimizing the number of storage cells required. For the Gill Method the above constants are

$$m = 1/2\ , \qquad r = 1 - \sqrt{1/2}\ , \qquad c_o = 1/6$$

$$n = 1/2\ , \qquad s = -\sqrt{1/2}\ , \qquad c_1 = (1/3)\ (1 - \sqrt{1/2})$$

$$p = 1\ , \qquad t = 1 + \sqrt{1/2}\ , \qquad c_2 = (1/3)\ (1 + \sqrt{1/2})$$

$$c_3 = 1/6$$

The Gill process further systematizes the calculation so as to increase the accuracy and simplify the coding.

<u>The Subroutine</u> As used in the Gill Method Subroutine, the process is as follows:

1st pass:

Advance x by $(1/2)h$

$$k_{io} = h \cdot f_i\ (x,\ y_{10},\ y_{20},\ \ldots,\ y_{no})$$

$$r_{il} = (1/2)k_{io} - q_{io}$$

$$q_{il} = q_{io} + 3r_{il} - (1/2)k_{io}$$

$$y_{il} = y_{io} + r_{il}$$

Calculate $f_i\ (x, y_{11}, y_{21}, \ldots, y_{n1})$ in programmer's own coding.

2nd pass:

$$k_{il} = h\ f_i\ (x, y_{11}, y_{21}, \ldots, y_{n1})$$

$$r_{i2} = (1 - \sqrt{1/2})\ (k_{il} - q_{il})$$

$$q_{i2} = q_{il} + 3r_{i2} - (1 - \sqrt{1/2})\ k_{il}$$

$$y_{i2} = y_{il} + r_{i2}$$

Calculate $f_i\ (x, y_{12}, y_{22}, \ldots, y_{n2})$ in programmer's own coding.

3rd pass:

Advance x by $(1/2)h$

$$k_{i2} = h \cdot f_1 (x, y_{12}, y_{22}, \ldots, y_{n2})$$

$$r_{i3} = (1 + \sqrt{1/2})(k_{i2} - q_{i2})$$

$$q_{i3} = q_{i2} + 3r_{i3} - (1 + \sqrt{1/2})\, k_{i2}$$

$$y_{i3} = y_{i2} + r_{i3}$$

Calculate $f_i (x, y_{13}, y_{23}, \ldots, y_{n3})$ in programmer's own coding.

4th pass:

$$k_{i3} = h \cdot f_i (x, y_{13}, y_{23}, \ldots y_{n3})$$

$$r_{i4} = (1/6)(k_{i3} - 2q_{i3})$$

$$q_{i4} = q_{i3} - 3r_{i4} - (1/2)k_{i3}$$

$$y_{i4} = y_{i3} + r_{i4}$$

Calculate $f_i (x, y_{14}, y_{24}, \ldots, y_{n4})$ in programmer's own coding.

Errors The paper by S. Gill mentioned previously includes a detailed analysis of errors, both truncation error and round-off error.

The expression for the truncation error in δy_i is too complicated to give here, but its dominating term, the author states, is

$$\frac{h^5}{-120} \sum_{\substack{o \\ j,k,l,m}}^{n} \left[f_j \frac{\partial f_k}{\partial y_j} \cdot \frac{\partial f_l}{\partial y_k} \cdot \frac{\partial f_m}{\partial y_l} \cdot \frac{\partial f_i}{\partial y_m} \right]_{x=X} \quad \text{where } y_o = x.\ f_o = 1,$$

and the truncation error in δy_i will be approximately this when the second partial derivatives are all close to zero. It is probably more useful to say merely that the truncation error is of the order of h^5.

The standard deviation in $y_i - (1/3)q_i$ over one step from all rounding off errors is (where f is the quantity mentioned in the section on notation)

$$1/6 \left[7/3 \left\{ 2^{-2f} + (1/16)h^2 \sum_j \left(\frac{\partial f_i}{\partial y_j} \right)_X^2 \right\} \right]^{1/2} u, \quad u = \text{the value of one unit in the last digit of y.}$$

Machine Checking

A driver routine solved two systems of equations both separately and concurrently, using the Gill Method Subroutine. The two systems solved are given below to indicate accuracy and to serve as examples.

1. Equations

$$\left.\begin{aligned} \frac{dy_1}{dx} &= y_2 \\ \frac{dy_2}{dx} &= -y_1 \end{aligned}\right\} \text{ equivalent to the second order equation, } \frac{d^2y}{dx^2} + y = 0.$$

$\Delta x = .0872664626 = \pi/36 = 5^\circ$

Initial Conditions

At $x=0$, $y_1 = 0$ and $y_2 = 1$.

Solution

$y_1 = \sin x$

Accuracy

In a spot check of the results, the greatest absolute error observed was 1.5×10^{-6}. (For $x = 3.1415925696$, $y_1 = .0000015425$. However, sin x = .000000084).

2. Equations

$$\left.\begin{aligned} \frac{dy_1}{dx} &= y_2 \\ \frac{dy_2}{dx} &= y_3 \\ \frac{dy_3}{dx} &= \frac{y_3 + 4x^2}{x} \end{aligned}\right\} \text{ Equivalent to the third order equation } x\frac{d^3y}{dx^3} - \frac{d^2y}{dx^2} = 4x^2$$

$\Delta x = .1$

Initial Conditions

At $x = .1$, $y_1 = .000025$, $y_2 = .001$, $y_3 = .03$

Solution

$$y_1 = \frac{x^4}{3} - \frac{x^3}{60} + \frac{x}{6000} - \frac{1}{120{,}000}$$

Accuracy

In a spot check of the results, the greatest relative error observed was 3.4×10^{-6}. (For $x = .1999999975$, $y_1 = .00042499858$. However, the solution is actually .00042500002).

```
D            GM     1024                                  02000  00 00000 00000
D           1GM     1065                                  02051  00 00000 00000
D           2GM    49880                                  63230  00 00000 00000
D           3GM    49921                                  63301  00 00000 00000
D           0GC     1083                                  02073  00 00000 00000
D           1GC    49939                                  63323  00 00000 00000
D           0GT       23                                  00027  00 00000 00000
2GM00   00 00000 00000                                    63230  00 00000 00000
2GM01   MJ                                                63231  45 00000 00000
2GM02   MJ          1GM00   TO SET UP                     63232  45 00000 02051
2GM03   MJ          0GM36   ENTER                         63233  45 00000 02044
2GM04   TP 0GM00    Q0000   RE ENTER                      63234  11 02000 10000
2GM05   QJ 0GM06    0GM40   WHAT                          63235  44 02006 02050
2GM06   QJ 0GM38    0GM07     PASS                        63236  44 02046 02007
2GM07   TP Q0000    0GM00   STORE CTR                     63237  11 10000 02000
2GM08   RA 0GM10    0GC01   UP PASS                       63240  21 02012 02074
2GM09   RP 30003    0GM11   STORE                         63241  75 30003 02013
2GM10   TP          0GT05    PASS CONS                    63242  11 00000 00034
2GM11   TU          0GM15   RESET                         63243  15 00000 02017
2GM12   TV          0GM31    ADDRESSES                    63244  16 00000 02037
2GM13   TP          0GT08   RESET N 1                     63245  11 00000 00037
2GM14   RP 30003    0GM16                                 63246  75 30003 02020
2GM15   TP          0GT0C                                 63247  11 00000 00027
2GM16   MP          0GT00   CALC K                        63250  71 00000 00027
2GM17   LA A                 SHIFT L                     63251  54 20000 00000
2GM18   TP A        0GT03   STORE K                       63252  11 20000 00032
2GM19   MP 0GT05    0GT03   AK                            63253  71 00034 00032
2GM20   MA 0GT06    0GT02   AK BQ                         63254  72 00035 00031
2GM21   LA A           38                                 63255  54 20000 00046
2GM22   TP A        0GT04   STORE R                       63256  11 20000 00033
2GM23   AT A         A      2R                            63257  35 20000 20000
2GM24   AT 0GT04     A      3R                            63260  35 00033 20000
2GM25   LA A           34                                 63261  54 20000 00042
2GM26   MA 0GT07    0GT03   3R CK                         63262  72 00036 00032
2GM27   LA A           38   ADD OLD Q                     63263  54 20000 00046
2GM28   AT 0GT02    0GT02   EQUALS Q                      63264  35 00031 00031
2GM29   RA 0GT01    0GT04   NEW Y                         63265  21 00030 00033
2GM30   RP 30002    0GM32   STORE                         63266  75 30002 02040
2GM31   TP 0GT01             Y AND Q                     63267  11 00030 00000
2GM32   RA 0GM15    0GC01   ADVANCE                       63270  21 02017 02074
2GM33   RA 0GM31    0GC02    ADDRESSES                    63271  21 02037 02075
2GM34   IJ 0GT08    0GM14   CYCLE N                       63272  41 00037 02016
2GM35   MJ                                                63273  45 00000 00000
2GM36   TU 0GC00    0GM10                                 63274  15 02073 02012
2GM37   TP 0GM21    Q                                     63275  11 02025 10000
2GM38   RA                                                63276  21 00000 00000
2GM39   MJ          0GM07                                 63277  45 00000 02007
```

PX 71900-9-(161)

```
2GM40   QJ 0GM01 0GM07                                  63300  44 02001 02007
3GM00   SP 0GM01 00015                                  63301  31 02001 00017
3GM01   TU A     1GM02                                  63302  15 20000 02053
3GM02   TP       0GT00                                  63303  11 00000 00027
3GM03   TV 0GT00 0GM35                                  63304  16 00027 02043
3GM04   TU 0GT00 1GM05                                  63305  15 00027 02056
3GM05   TV       0GM17                                  63306  16 00000 02021
3GM06   RA 0GT00    15                                  63307  21 00027 00017
3GM07   RP 10003 1GM09                                  63310  75 10003 02062
3GM08   TU 0GT00 0GM11                                  63311  15 00027 02013
3GM09   RA 0GM13    15                                  63312  21 02015 00017
3GM10   AT    15 A                                      63313  35 00017 20000
3GM11   TU A     0GM16                                  63314  15 20000 02020
3GM12   TU A     0GM38                                  63315  15 20000 02046
3GM13   LQ A        21                                  63316  55 20000 00025
3GM14   TV Q      0GM38                                  63317  16 10000 02046
3GM15   RA 0GM38    15                                  63320  21 02046 00017
3GM16   RA 0GM01    16                                  63321  21 02001 00020
3GM17   MJ       0GM01                                  63322  45 00000 02001
1GC00      GC                                           63323  00 02073 00000
1GC01         3      B                                  63324  00 00003 00000
1GC02                3 B                                63325  00 00000 00003
1GC03    5              - 1  34  A1 SCALE 34            63326  10 00000 00000
1GC04   -1                   34  B1       34            63327  57 77777 77777
1GC05   -5              - 1  34  C1       34            63330  67 77777 77777
1GC06   02 92893 21881 - 1  34  A2 SCALE 34            63331  04 53730 31460
1GC07   -2 92893 21881 - 1  34  B2       34            63332  73 24047 46317
1GC08   -2 92893 21881 -01  34  C2 SCALE 34            63333  73 24047 46317
1GC09    1 70710 67812      34  A3       34            63334  33 24047 46320
1GC10   -1 70710 67812      34  B3       34            63335  44 53730 31457
1GC11   -1 70710 67812      34  C3       34            63336  44 53730 31457
1GC12    1 66666 66667 - 1  34  A4       34            63337  02 52525 25253
1GC13   -3 33333 33333 - 1  34  B4       34            63340  72 52525 25252
1GC14   -5              - 1  34  C4       34            63341  67 77777 77777
```

PX 71900-9-(161)

Pseudo-Random Number Generator Subroutine

by Harold Dahlbeck

The basic method used in this subroutine was described by D. H. Lehmer in "The Annals of the Computation Laboratory of Harvard University", Volume XXVI, Proceedings of a Second Symposium of Large-Scale Digital Calculating Machinery, Harvard University Press, Cambridge, Massachusetts, 1951.

The method itself requires the following simple form to be used as the iteration formula:

$$X_{i+1} = CX_i \pmod{p} \qquad i = 1, 2, 3, \text{---}$$

Although this method is considered to be quite optimal in guaranteeing both maximum randomness and greatest periodicity of the X_i, there are considerable problems involved in the obtaining of C and p subject to the following conditions:

1. p should be the largest prime number which can be contained in a register.
2. C must be of the form K^L where K is required to be a primitive root of p and L has to be relatively prime to p-1. In addition C should be as large as possible. The following two numbers satisfy the above conditions.

$$p = 2^{35} - 31 = 34,359,738,337$$

$$C = 5^{13} = 1,220,703,125$$

A subroutine for generating the X_i is given below. The periodicity of the X_i will be $p - 1 = 2^{32} - 32 = 34,359,738,336$. Any positive number may be used as a starting point for the series.

1103 Subroutine for Pseudo-Random Numbers

01000	71	01777	01776	$5^{13}.R_1$ A
01001	73	01775	10000	$(5^{13}.R_1) \bmod p$ A
01002	11	20000	01777	A mod p R_1
01775	37	77777	77741	$2^{35} - 31$
01776	01	10604	71625	5^{13}
01777	00	00000	00001	R_i

PX 71900-9-(162)

WF-163

CENTRAL EXCHANGE INDEX:

TITLE: Line Printer Decimal Output

ENTRIES IF MORE THAN ONE OR NOT STANDARD:

STANDARD: YES x NO

CODING CHECK: x BY C. S. Fluke

MACHINE CHECK: x BY C. S. Fluke

SELF-RESETTING: YES x NO

1. Initial location of argument: Q_u=lst address of input: Q_v=(no. of digits)$_8$ =(1-92)$_{10}$

   Final location of argument: 00011

   Location of the function: paper

2. ADDRESSES:

   (a) Instructions: 01000 through 01065

   (b) Constants and temporary storage: 01066 through 01134

   (c) Constant pool used: 00040, 00057, 00073, 00074

   (d) Temporary storage pool used: 00010 - 00017

3. INITIAL SETTING OF X:

   (a) Range on x:

   (b) Scaling of x and f(x):

   (c) Brief description of numerical method:

4. ACCURACY: Full 100%

5. ALARM-CONDITIONS FOR OUT-OF-RANGE TEST: For illegal code:

   A_{R_u}= illegal code; A_{R_v} = place in word of input = (1-11)$_8$

PX 71900-9-(163)

A_{L_u} = address of code word.

6. SPEED: Line printer limited

7. MISCELLANY: Program will take parameters defining input, make card image, and print one line starting at left most digit. Paper is advanced by format switches for col. #1.

Input must have information for the number of digits to be printed, including codes for spaces,-,., as well as the decimal digits. These codes are stored in the input as hexadecimal digits, i.e., 4 bits per digit, and are packed 9 digits per machine word, going from left to right in adjoining registers.

The codes can be derived as the immediate result of a preceding conversion routine. They are:

Digit to be printed	8421 binary code
space	1111
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
- (odd cols.) . (even cols.)	1010

EV PROGRAM CN 007

Table of Contents

ANALYSIS
PREPARED BY Donn Parker
CHECKED BY . Charles Swift
REVISED BY Matt Vuletich

CONVAIR
A DIVISION OF GENERAL DYNAMICS CORPORATION
SAN DIEGO

EV PROGRAM CN 007

1103 PROGRAM FOR COMPUTING EIGENVALUES AND EIGENVECTORS OF REAL, SYMMETRIC MATRICES

1. INTRODUCTION:

This program is designed to determine all vectors y_i and all scalars λ_i which together with a given real, symmetric matrix A of order $N \leq 64$ satisfy the relationship $Ay_i = \lambda_i y_i$. The mathematical technique employed is based on the Hestenes-Karush gradient methods[1]. The program performs the Hestenes-Karush near-optimum gradient method[2].

Any number of the eigenvectors and their associated positive, non-zero eigenvalues may be obtained in decreasing sequence with respect to algebraic magnitude of eigenvalues. Extremely close eigenvalues and eigenvalues close to zero with respect to the total range of eigenvalues slows convergence and in extreme cases prevents convergence.

The program is made to take advantage of a matrix with large numbers of zero elements by representing it in dehydrated form ie. with blocks of zero elements replaced by flags indicating the number of zeros removed. For larger matrices ($40 < N < 64$), this program becomes more efficient with respect to time of solution and accuracy.

Ten decimal digits, unpacked floating point number representation is used (See CA 001 report ZM 491).

11. EQUATIONS:

The basic formuli may be summarized as follows. Given an approximation x_i, define x_{i+1} by: $x_{i+1} = x_i + \alpha_i \xi_i$

where $\xi_i = Ax_i - \mu(x_i)x_i$

and $\mu(x_i) = x_i^* Ax_i / x_i^* x_i$ (The Rayliegh Quotient)

and $\alpha_i = \beta / |\mu(x_i) - \mu(\xi_i)|$ $0 < \beta \leq 1$

When x has converged to the eigenvector y, $\mu(x)$ has converged to the eigenvalue λ. The convergence to the eigenvalue proceeds twice as fast as convergence to the eigenvector. The program uses $\beta = .8$, but this can be varied easily from iteration to iteration. Each eigenvector is obtained by forcing the iterate x_i to remain in the orthogonal compliment of the subspace spanned by the eigenvectors already obtained (a symmetric, real matrix has mutually orthogonal eigenvectors). This is accomplished by applying the following orthogonalization correcting formula every few iterations.

$\bar{x}_i = x_i - \sum_{j=0}^{p-1} [x_i^* y_j / y_j^* y_j] y_j$ where x_i is the i th iterate converging to the eigenvector y_p and y_j $(j=0,1,\ldots,p-1)$ are the p mutually orthogonal eigenvectors thus far obtained.

An arbitrary method of determining convergence is used as follows. The iterate x_i and the associated Rayliegh quotient $\mu(x_i)$ are assumed to have converged to the eigenvector y and the eigenvalue λ respectively with d bits of accuracy when $\min_k \left\{ \text{exponent } [(Ax_i)_k] - \text{exponent } [(\xi_i)_k] \right\} \geq d, k = 1,2,\ldots,N$ where $(Z_i)_k$ is the k th element of the vector Z_i.

111. PROGRAM CONSTANTS AND TERMINOLOGY:

DESCRIPTION	SYMBOL	PERMANENT LOCATION	TEMPORARY
Order of matrix	N	40005	00005
Number of eigenvectors in the orthogonal compliment of converging iterate	E	40004	00004
Number of eigenvectors desired	E	40003	00003
Bits of accuracy attained	d		00012
Desired bits of accuracy	T	40564-40576	01624-01636
Number of vectors desired in output	M		00010
Number of cells in ES working storage	K-	40406	00406
First vector in ES	V_1	00420 to 00417 + 2N	
Second vector in ES	V_2	00420 + 2N to 00417 + 4N	
Vector in MD	VMD	41252 to 41451	
Matrix in MD	AMD	41452 to 53377	
Operands for floating arithmetic	OP1,OP2		00025-00030
Result	Res.		00031,00032
See section VI,H.	α index	40247	00247
Counts interations	Iteration index		00047
Counts iterations per accuracy test	Test index	40734	01774
Counts accuracy tests per type out	Typing index	40561	01621
Counts iterations per orthogonalizatior	Orthogonalization index	41035	00305

PX 71900-9-(164)

ANALYSIS
PREPARED BY Donn Parker
CHECKED BY Charles Swift
REVISED BY Matt Vuletich

CONVAIR
A DIVISION OF GENERAL DYNAMICS CORPORATION
SAN DIEGO

IV. PROGRAM CHARACTERISTICS

The program will handle all matrices of order $N \leq 64$, but there is storage space in MD for only storing an undehydrated matrix of order $N \leq 50$ ie. $(11726)_8$ cells capable of storing $(4753)_8$ two register floating elements (See CA 001 report ZM 491). However, the storage region may be expanded (See appendix b.).

The program as presented is set up to test the accuracy of convergence every other iteration, to type the value of d every other accuracy test ie. every fourth iteration, to orthogonalize every sixth iteration, to compute a new α every other iteration, and to cycle through three values of β (all set equal to .8) during iterations.

Standard subroutines used and included on EV paper tapes.

CA 001 Two Register Single Pricision Floating Point Arithmetic Package.

IC 003 Decimal to Two Register Floating Binary Card Input. This has been modified to read dehydrated matrices.

IC 004 Floating Point Card Output. This has been assembled and stored at 76600.

IB 002 Alarm, Octal, and Flexprint Package.

EV SUBROUTINES

MT Dump and Restore

Inner Product

Dehydrated Matrix-vector Multiplication

Rayliegh Quotient

Orthogonalization

$V_2 + SV_1$

PROGRAM PARTS:

Starter.

Check sum taken and typed. Program is set up for a given order matrix, initial eigenvector iterates are stored, the complete contents of the drum and ES is stored on MT, and the eigenvalue and eigenvector region is stored on MT following the complete MD, ES dump. The starter is destroyed when computation starts.

Part I. d is computed, typed, and compared with T. Part I is stored on MD and is transferred to ES only when needed.

Part II. Iteration routine. It remains in ES except when orthogonalizing. Part II is restored to ES from MD for each eigenvector.

Part III. Computed eigenvector and eigenvalue are stored on MD, program is set up for the next eigenvector iteration, final output is controlled. Part III is stored on MD and is transferred to ES only when needed.

MT CONTROL: Eigenvectors, eigenvalues and inner products of eigenvectors are stored on MT. Program, eigenvectors, eigenvalues, and inner products are restored from MT to MD and ES. This acts as a service routine and is under control of the machine operator.

MT Photo Dumps: After the program is set up for a given matrix, the starter dumps MD and ES onto MT #0 and the initial eigenvectors and eigenvalues region onto MT# 0 and MT# 2. From time to time (about every half hour), at the operators discretion, the eigenvectors and eigenvalues may be dumped on MT# 0 and MT# 2. Then in case of a failure the whole program and eigenvectors and eigenvalues from the last dump may be restored to MD and ES and the problem continued from the point of the last MT photo dump. By selecting or not selecting MT#2 the eigenvectors and eigenvalues are restored from either MT# 2 or MT# 0 respectively. Having the

eigenvectors and eigenvalues stored on two MT units reduces the possibility of losing the information.

The MT dumping routine takes three minutes forty seconds to dump MD and ES and one minute ten seconds to restore MD and ES. It requires $(1017)_8$ blocks to store MD and ES and $\left\{37\left(\left[\frac{2NE+400}{1737}\right]+1\right)\right\}_8$ blocks to store the eigenvector and eigenvalue region where [ ] indicates "the integer part of". Included in the eigenvector and eigenvalue region are stored the inner products (dot products) of each eigenvector with itself (These are used in the orthogonalization process).

EXTERNAL CONTROLS: There are two basic controls that are used during the computational part of the program. MS# 2 at 01535 causes a stop before setting up for the next eigenvector solution. E + 1 appears in Q and E appears in A. By changing (Q) or (A) at this point the eigenvector to be converged upon and the number of eigenvectors desired may be changed. MS# 1 at 01704 in Part I of the program after one or several iterations (depending on the value of the Test Index) causes a stop with T in Q and d in A. By changing (Q) the desired accuracy T may be altered. If T is changed, the new value is typed on the supervisory control typewriter when the computer is restarted.

At an MS# 3 stop (there are several of them) at 00241 the mantissa of β (exponent = 0) appears in Q and may be altered for the next iteration only.

All other external controls may be found in the operating instructions.

V. INPUT - OUTPUT

INPUT is performed using the Card Input paper tape which contains IC 003 with a change to accomodate reading in the dehydrated matrix. Input consists of the dehydrated matrix punched on cards by rows, one row immediately following the preceding row. The card format and header card are described in IC 003.

The EV program may be performed without this Input operation as long as the matrix is stored in single precision floating point form (see CA 001) by rows at the address 41452.

DEHYDRATED MATRIX: One zero element is represented as true zero. Several Zero elements in sequence (row end intervening or not) are represented as a true zero representing the first zero except that the exponent is the number of zero elements following the first zero element. See the sample problem for an example.

After E eigenvectors have been obtained the program automatically exits to the output routine where an MT dump is performed, and all eigenvalues are punched in floating decimal form on cards (see IC 004). When the computer stops, m, the number of eigenvectors wanted starting with the last one obtained, is inserted, and when the computer is started they are punched in floating decimal form on cards in the order in which they were obtained. The computer then exits to an MS #2 stop.

SUPERVISORY CONTROL TYPEWRITER OUTPUT: At the beginning of an iteration process the SCT types the value of E and T. Every fourth iteration the iteration count and d are typed. When T is manually changed, the new value of T is

typed. When the process has converged (d>T), one more iteration is performed, iteration count and d are typed, and the first three octal digits of the mantissa and last three octal digits of the exponent of the eigenvalue are typed.

Whenever a MT dump of eigenvectors and eigenvalues is performed, "MT" is typed.

VI. OPERATING INSTRUCTIONS:

A. Paper tape start for initial starting of a problem.

1. It is advisable to clear MD to all zeros and put any convenient service routines in the 70000 - 77777 part of MD.
2. MT #0, MT #2 should be set to the desired locations.
3. Matrix Input. (Optional. See section V of this report.)

   Load paper tape, " Card Read IC 003 revised".

   Load matrix deck of cards (with header card) in read hopper.

   MD start 00300. Reads cards, types check sum and "done".
4. Load main paper tape, "EV".
5. MD start at 40000.
6. At MS #0 stop, 01017 put N x 2^0 → (Q), E x 2^0 → (A).
7. Start. MT dump requires about three minutes forty seconds. Stop at MS #0 1161.
8. Start. Iteration starts.

B. Output. When E = E, computer stops at 01572.

1. Number of vectors wanted = m x 2^0 → (Q).
2. Start. Stop at 01613 after output, and MT dump.

C. Intermediate MT Dump.

1. At MS #2 stop 01535 Set 76430 → (PAK)

2. Start (MS #2 on). Stops MS #0 at 01532.

3. Start. MS #2 stop 01535. Now ready to continue problem or leave computer to start problem from this point at another time.

D. Continuation from MT start.

1. Set MT #C, MT #2 as in step A-2.

2. Load paper tape "MT Dump and Restore".

3. MS #2 on. MJ #2 { on, restore from MT # 2 / off, restore from MT # 0

4. MD start at 76370. Stops at MS # 0, 01532.

5. Start. Stops at MS # 2, 01535 with E x 2° = (Q), E x 2° = (A) ready to continue problem.

E. Intermediate start from MD.

1. MS #2 on.

2. MD start at 40000. MS #2 stop 01535. Now ready to continue problem from last eigenvector obtained.

F. Trouble. Trouble can arise from one of two causes, either computer failure or operator's error. Trouble shows up as a computer fault (SCC, MCT, etc.) or as a failure to converge as shown on the typewriter output. In case of trouble do the following.

1. Perform step E. If the last eigenvector obtained cannot be converged upon again, then perform the next step.

2. If MT # 0 has not been set back to its origin, back tape $(1017)_8$ blocks and do step D, omitting substep 1. Otherwise just do step D.

3. If step D fails (ie. MT dump has been destroyed), it may be possible to start from paper tape (step A) and instead of dumping the eigenvectors and eigenvalues on MT, restore them. Do step A with MS # 2

and MJ # 2 on and with (41262) = 75 30210 76370 and (76463) = 37 76463 76464.

To restore just the eigenvectors and eigenvalues from MT # 0 or MT # 2, perform step D with the following changes.

After substep 1. Advance MT # 0 $(1017)_8$ blocks.

Replace substep 4 with "MD start at 76374 with 2 NE x 2° in A".

If MT # 0 dump is gone, repeat the above steps using MT # 2.

ALARMS: Alarm print (see IE 002) at 01565 indicates $E > \bar{E}$. Start computer and at MSC stop 01540, E = (A). Correct $\bar{E}$ and start. Computer jumps to $E = \bar{E}$ test. Alarm print any place else indicates attempted division by zero.

G. Convergence control. Bits of accuracy will usually be negative for a number of iterations and when positive, they will fluctuate, sometimes quite violently. The technique is to stop the iteration process after as few iterations as possible with the most accuracy. This is accomplished using the MS # 1 stop and changing the desired accuracy to fit the case. See the sample typewriter output.

Bits of accuracy required have been prestored in the program (40564 to 40576), six values of T per word in order from right to left. When E equals some value, say E', and it is desired to go back to improve some previously obtained eigenvector ie. when E was equal to E' - S, then to return to the case E = E', all eigenvectors from E' - S to E' must be iterated upon to preserve orthogonalization.

H. Optional operations.

1. When starting from paper tape, MJ # 2 on eliminates storing all ones in the eigenvector region as the initial iterates.
2. MJ # 3 on avoids dumping the program on to MT, but does not

eliminate dumping the eiginvectors and eigenvalues region on MT # 0 and MT # 2.

3. At MS # 3 stop 00241 the mantissa of β is in Q and can be changed by altering (Q).

---

1. Hestenes, M.R. and Karush, W. A Method of Gradients for Calculation of Characteristic Roots and Vectors of A Real Symmetric Matrix, J. Research NBS 47, 5, (1951) RP 2227.

2. Stein, M.L. Gradient Methods in the Solution of Systems of Linear Equations, J. Research NBS 48, 6, (1951) RP 2330.

APPENDIX A: History and Operating Experience.

The eigenvector-eigenvalue problem for large, real, symmetric matrices with many zero elements is necessary for computing the mode shapes and frequencies of vibrations of multispar wings. The EV program for this purpose was first coded in January, 1955 by Dr. Marvin Stein and Donn Parker, and a preliminary report, CN 001 of Report Z M 491, was written in February, 1955. This first program has been used to solve problems involving 37 x 37, 35 x 35, 33 x 33, 59 x 59, 62 x 62, and 29 x 29 matrices. The program was recoded in December, 1955 by Donn Parker and is presented in this report.

The matrices we have had experience with have a zero eigenvalue and a large negative eigenvalue which made it impossible to use the conjugate gradient method to minimize and obtain the smallest eigenvalue first. We are interested in only the smallest ten eigenvalues (not including the zero and negative values) and their associated eigenvectors, but in order to obtain them we must first solve for all the larger values. Investigations are being carried on now to find a better, more direct method.

Provision has been made to skip the storage of the initial iterates all of whose elements equal one. If better approximations to the eigenvectors are known, they may be placed in the eigenvector region at the same time as the matrix input.

If the convergence is very slow for a particular eigenvalue, increasing the frequency of orthogonalization often helps. This may be done by manually changing the orthogonalization index or can be more easily done by lowering T so that the convergence is completed with very low accuracy, then start convergence on the same eigenvector thus starting with the iterate last obtained. This often "shakes loose" the iterate from a stagnating convergence, and it is

the result of orthogonalizing twice without any intermediate iterations.

Another cause of slow convergence is lack of accuracy of eigenvectors previously obtained. Convergence can be improved often by improving the accuracy obtained for previously found eigenvectors.

APPENDIX B: Listed here are all references in the program to the matrix, eigenvalues, eigenvectors, and inner products storage areas on the MD. The storage assignments for these areas may be changed by altering these references.

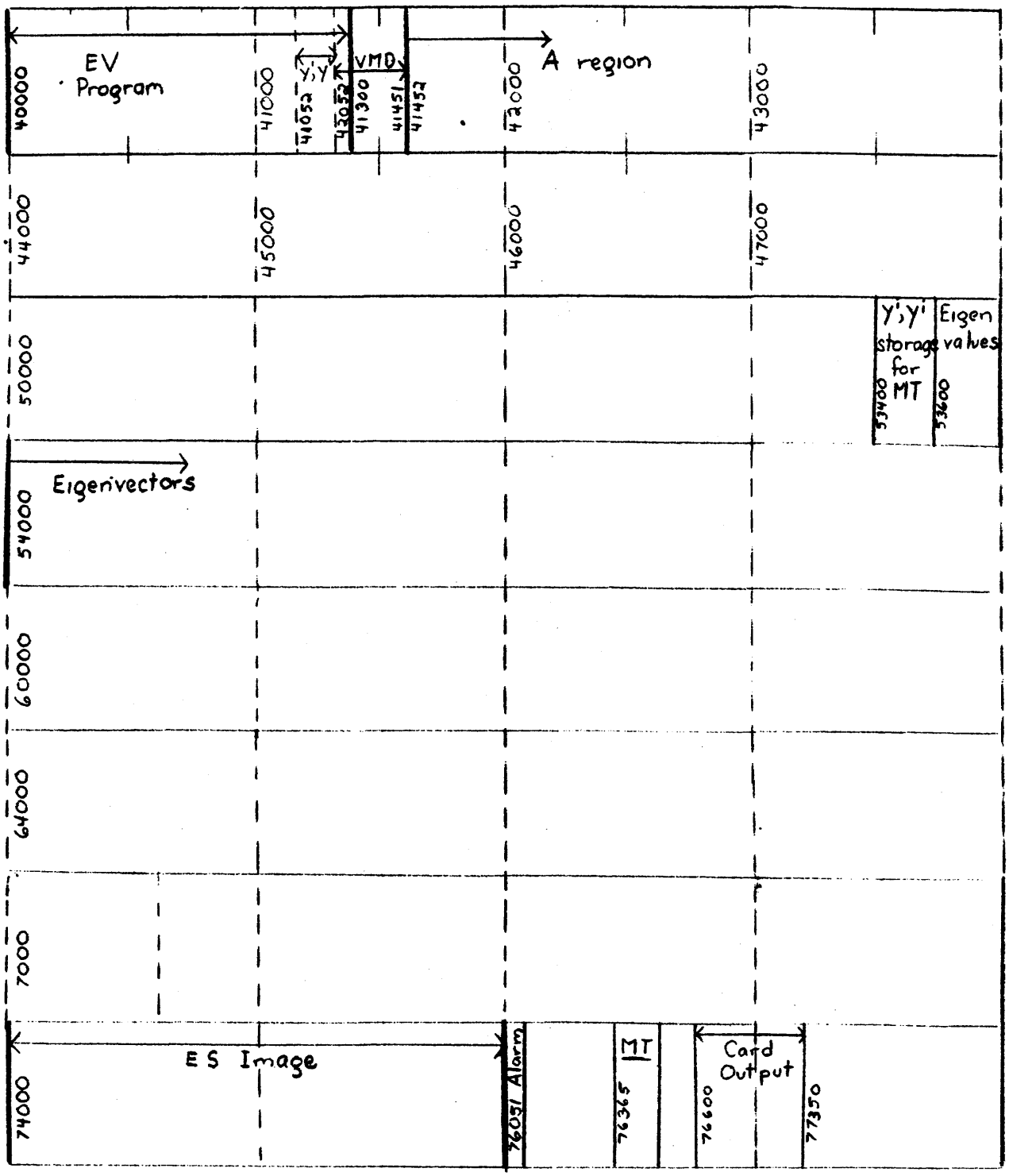
41452	Matrix A	Ref: 40015, 40016
53400	y,y	76423, 76425, 76471
53600	Eigenvalues	41266
54000	Eigenvectors	41250, 41252, 41253, 41265

ANALYSIS
PREPARED BY Donn Parker
CHECKED BY Charles Swift
REVISED BY Matt Vuletich

CONVAIR
A DIVISION OF GENERAL DYNAMICS CORPORATION
SAN DIEGO

PAGE CN 007-15
REPORT NO. ZM 491
MODEL All
DATE 6/20/56

MD Storage Assignments



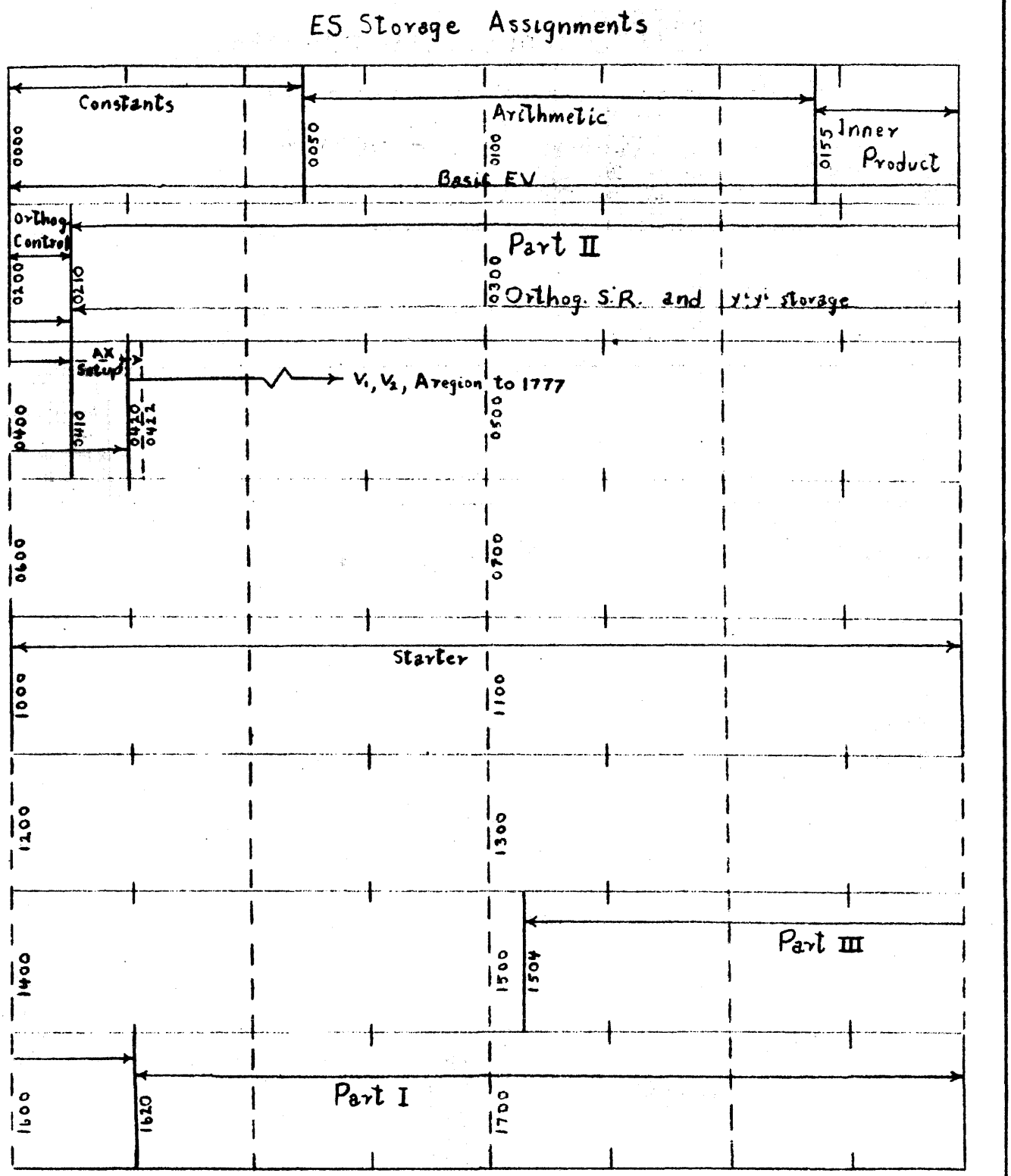
PX 71900-9-(164)

FORM 1512-A

ANALYSIS
PREPARED BY Dean Parker
CHECKED BY Charles Swift
REVISED BY Matt Vuletich

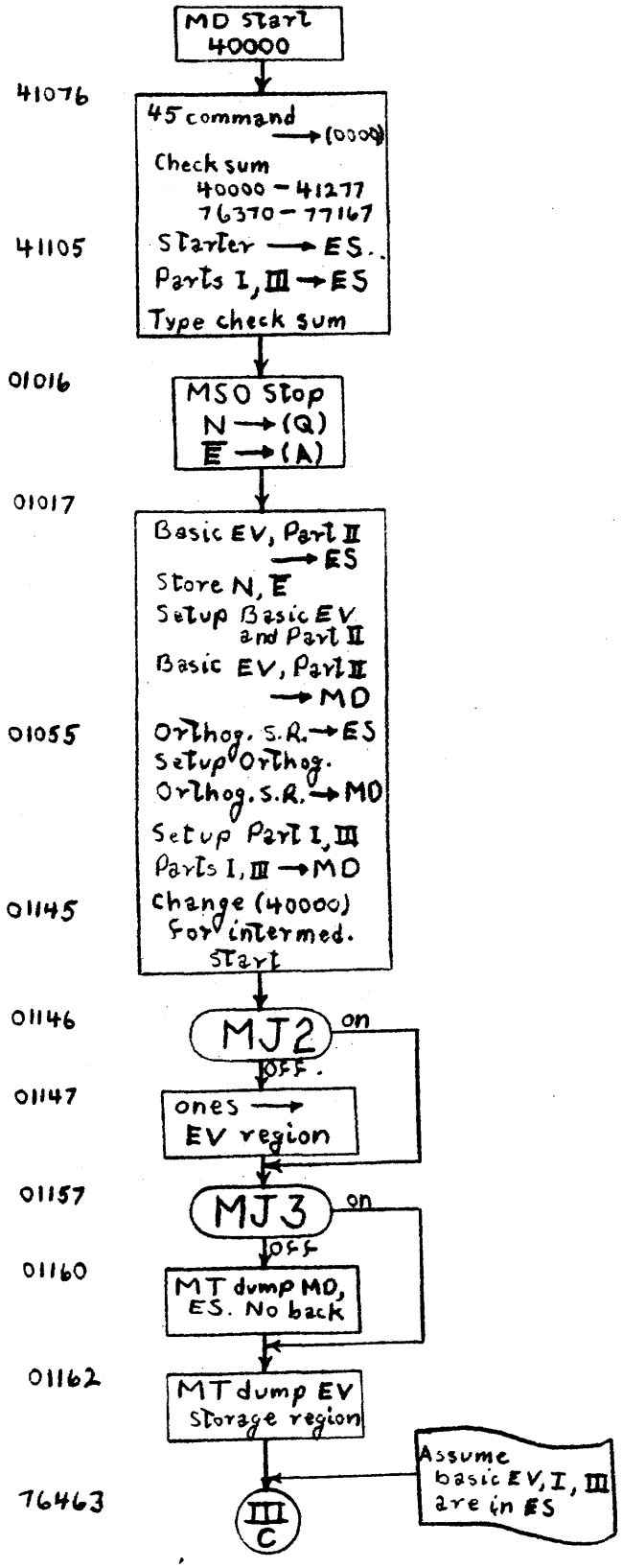
CONVAIR
A DIVISION OF GENERAL DYNAMICS CORPORATION
SAN DIEGO

PAGE CM 007-16
REPORT NO. ZM 491
MODEL All
DATE 6/20/56



PX 71900-9-(164)

EV Starter

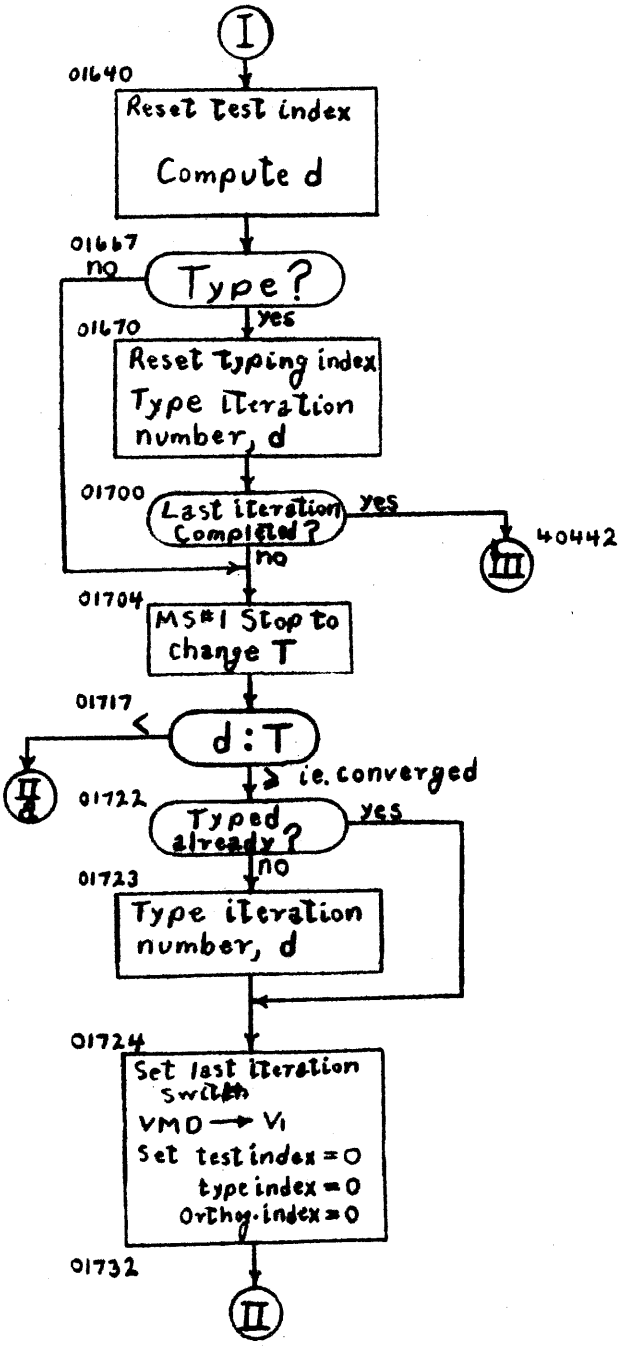


PX 71900-9-(164)

ANALYSIS
PREPARED BY Donn Parker
CHECKED BY Charles Swift
REVISED BY Matt Vuletich

CONVAIR
A DIVISION OF GENERAL DYNAMICS CORPORATION
SAN DIEGO

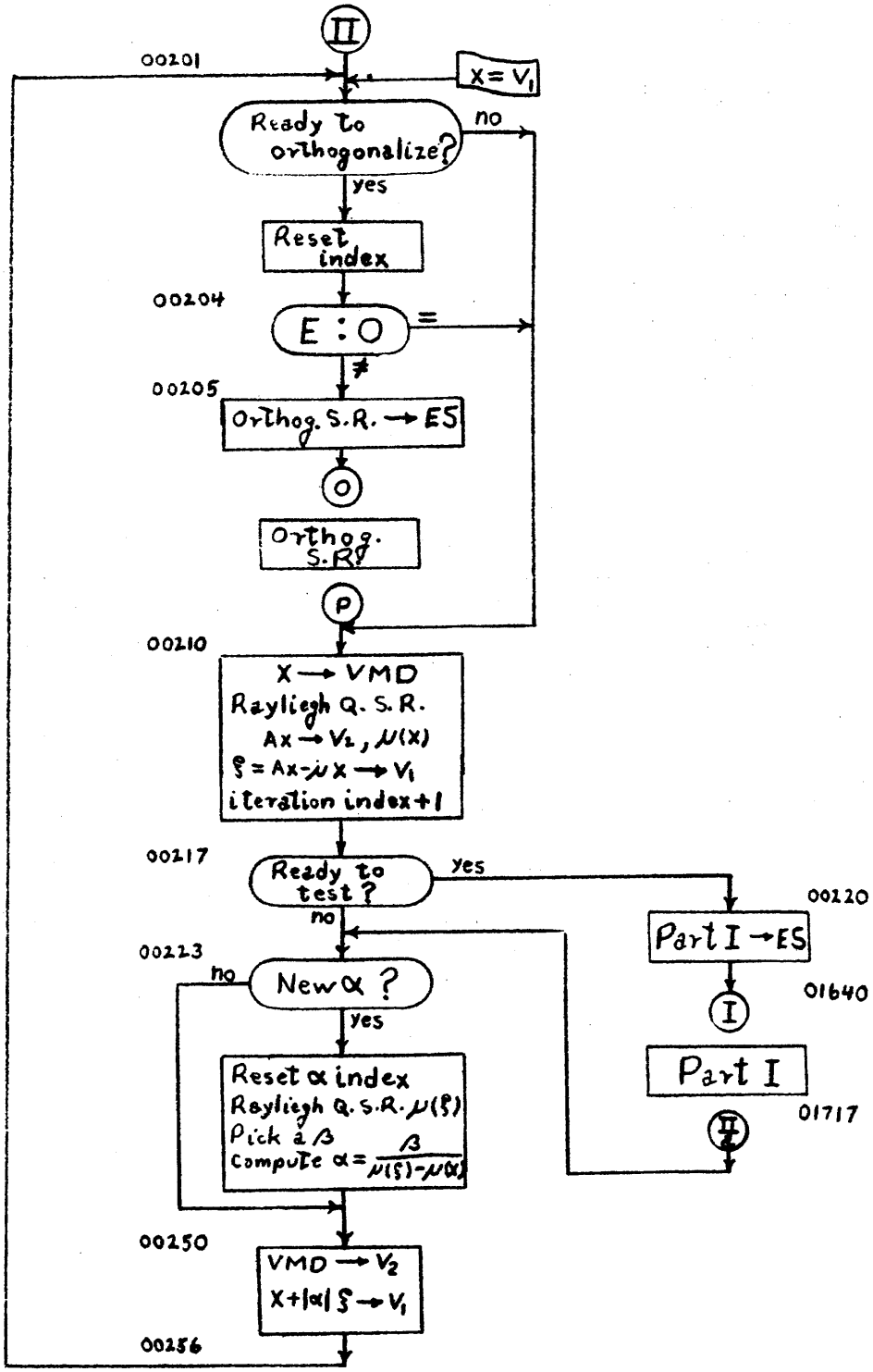
EV Part I



ANALYSIS
PREPARED BY Donn Parker
CHECKED BY Charles Swift
REVISED BY Matt Vuletich

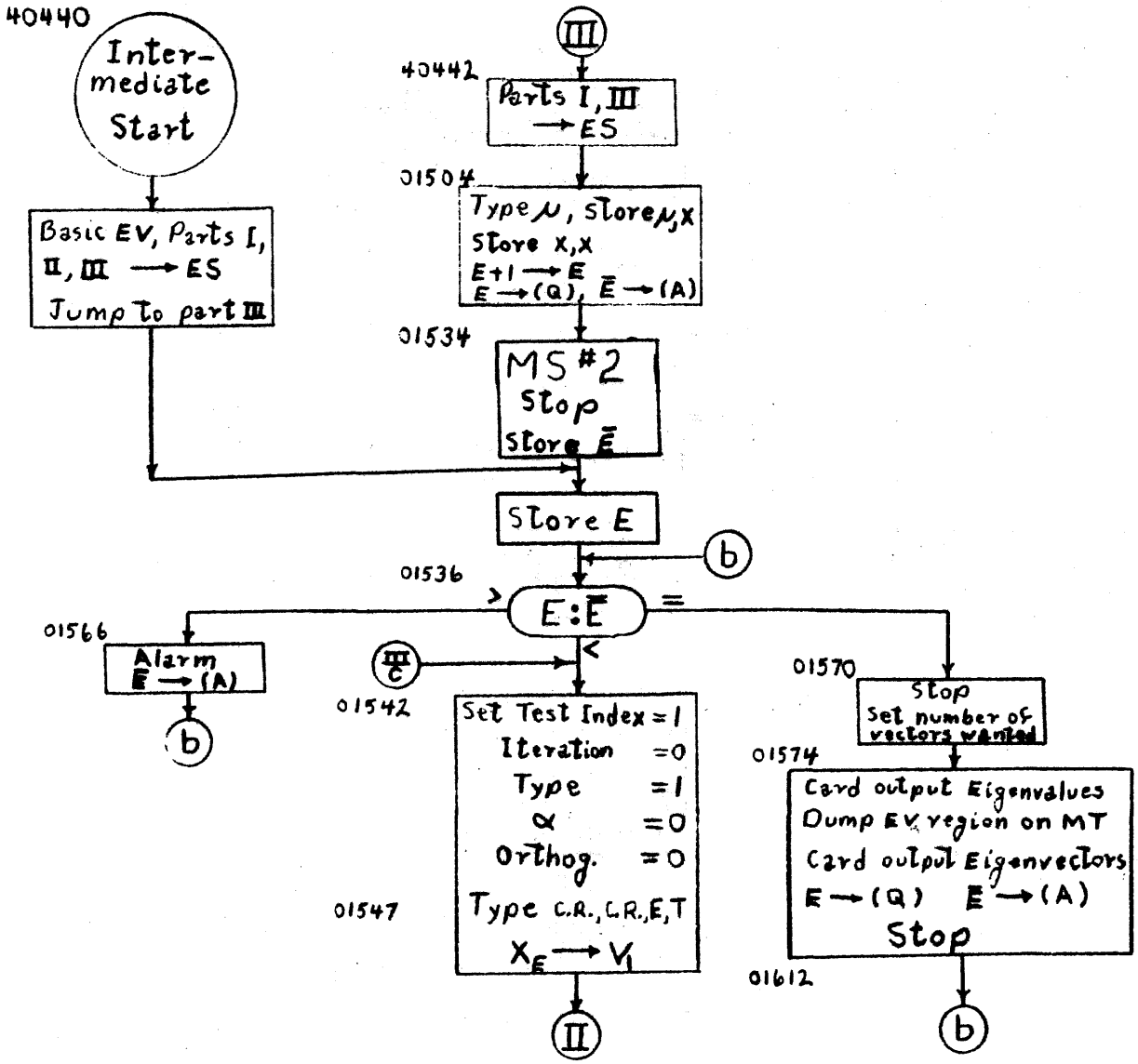
CONVAIR
A DIVISION OF GENERAL DYNAMICS CORPORATION
SAN DIEGO

EV Part II



PX-71900-9-(164)

EV Part III

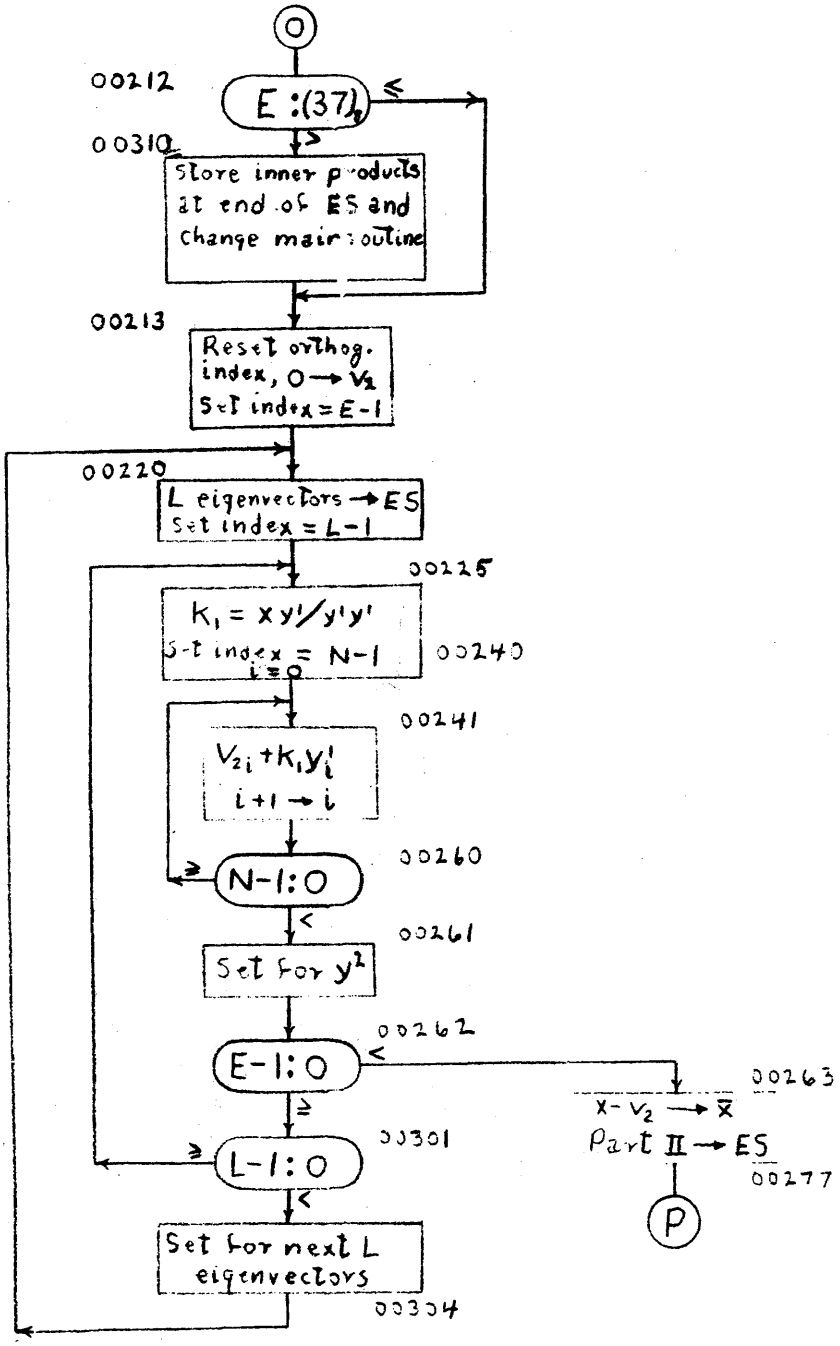


PX 71900-9-(164)

ANALYSIS
PREPARED BY Dann Parker
CHECKED BY . Charles Swift
REVISED BY Matt Vuletich

CONVAIR
A DIVISION OF GENERAL DYNAMICS CORPORATION
SAN DIEGO

EV Orthogonalization



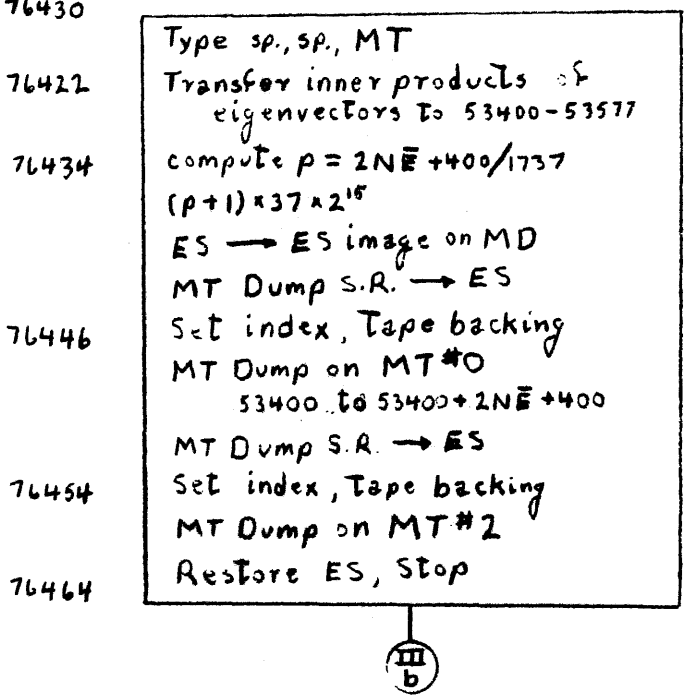
PX 71900-9-(164)

ANALYSIS
PREPARED BY Donn Parker
CHECKED BY • Charles Swift
REVISED BY Matt Vuletich

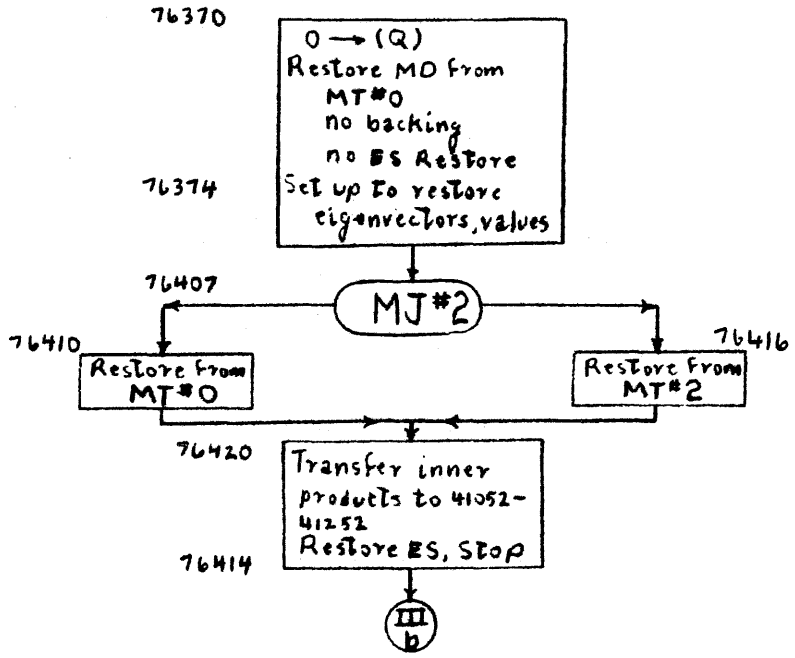
CONVAIR
A DIVISION OF GENERAL DYNAMICS CORPORATION
SAN DIEGO

MT Routines

EV MT Dump



Restore Problem From MT



SAMPLE SCT OUTPUT

for the following sample problem

```
671770371661                          EV check sum of matrix
done2344777361O5  MT                  EV check sum of program
                                          MT dump taken

000 042                               E=0, T=42

004 765                               4th iteration, no accuracy
010 771
014 000
020 004
024 006
030 012
034 014
040 017
044 033
050 033
054 041
056 042                               46th iteration, d=42
057 042 270 027                       Converged.  Eigenvalue

001 040                               E=1, T=40

004 767
010 767
014 776
020 005
024 011
030 020
034 022
040 025
044 026
050 030
054 031
060 034
064 036
070 036
074 035
100 035 036                           T changed to 36
104 035
110 034
114 035
120 040
121 040 223 027  MT                    MT dump after two vectors
                                        have converged
```

Sample Problem

"29 x 29 Matrix A"

101	.3139500675	4	.5048596477-	4	.6178035335	4	.6385646884-	4	.1515033341	4
106	.4309664059-	3	.1144258462	3	.0000000000		.9661205555-	1	.0000000000	0
111	.0000000000	0	.0000000000		.0000000000		.0000000000		.0000000000	0
116	.0000000000	0	.0000000000		.0000000000		.0000000000	0	.0000000000	0
121	.0000000000	0	.0000000000		.0000000000		.0000000000	0	.0000000000	0
126	.0000000000	0	.0000000000		.0000000000		.0000000000	0	.5048596477-	4
202	.1135023727	5	.2435556348-	5	.3472949305	5	.8239782252-	4	.2343888578	4
207	.6223256157-	3	.0000000000	*	.5254421003	2	.0000000000	0	.0000000000	0
212	.0000000000	0	.0000000000		.0000000000		.0000000000	0	.0000000000	0
217	.0000000000	0	.0000000000		.0000000000		.0000000000	0	.0000000000	0
222	.0000000000	0	.0000000000		.0000000000		.0000000000	0	.0000000000	0
227	.0000000000	0	.0000000000	0	.0000000000		.6178035335	4	.2435556348-	5
303	.1420318524	6	.3699632856-	6	.1475097189	6	.4196061725-	5	.1114095918	5
308	.0000000000	0	.9406537104-	3	.5240639171-	5	.3301825977	4	.1931808165	3
313	.0000000000	0	.0000000000	*	.2399646300	5	.0000000000	0	.1062027047-	4
318	.0000000000	0	.0000000000	0	.7200680527-	4	.1930041152	4	.0000000000	0
323	.0000000000	0	.2103429817-	3	.0000000000		.0000000000	0	.0000000000	0
328	.9019303291-	1	.0000000000	0	.6385646884-	4	.3472949305	5	.3699632856-	6
404	.2012897506	7	.9757463285-	6	.6139115888	6	.1629996030-	6	.0000000000	0
409	.1376238604	5	.5430837130	4	.1153271155-	7	.0000000000	0	.0000000000	0
414	.0000000000	0	.0000000000		.3481976210	6	.0000000000	0	.0000000000	0
419	.0000000000	0	.8001611163-	5	.0000000000		.0000000000	0	.0000000000	0
424	.0000000000	0	.0000000000		.0000000000		.0000000000	0	.0000000000	0
429	.0000000000	0	.1515033341	4	.8239782252-	4	.1475097189	6	.9757463285-	6
505	.1087084987	7	.7908157834-	6	.2528029819	6	.4172939739-	4	.2114887918-	5
510	.1819402783	1-	.0000000000	0	.1045721812-	7	.1425800171	4	.0000000000	0
515	.1024655024-	0	.0000000000	0	.3579432551	6	.0000000000	0	.0000000000	0
520	.1143933398	0	.3348422218-	1	.0000000000	*	.0000000000	0	.1139380806	0
525	.0000000000	0	.0000000000	*	.0000000000		.1611736022-	1-	.0000000000	0
601	.4309664059-	3	.2343888578	4	.4196061725-	5	.6139115888	6	.7908157836-	6
606	.2930202641	5	.1330461241-	7	.1033342696	6	.3152509181	6	.0000000000	0
611	.0000000000	0	.2273862205	4	.6737892793-	6	.1050351686	5	.0000000000	0
616	.0000000000	0	.0000000000	*	.3313806204	6	.0000000000	0	.0000000000	0
621	.0000000000	0	.1196194835-	6	.0000000000	*	.0000000000	0	.1478294679	5

RA 11700-Y-(164)

"29 x 29 Matrix A"

626	.0000000000	0	.0000000000	0	.3527649574-	4	.0000000000	0	.1144258462	3
702	.6223256157-	3	.1114095918	5	.1629996030-	6	.2528029819	6	.1330461241-	7
707	.8233909877	6	.1054048388-	6	.2442203826-	6	.0000000000	0	.0000000000	0
712	.0000000000	0	.1855177321	4	.1629515823-	6	.0000000000	0	.0000000000	0
717	.0000000000	0	.0000000000	0	.5834576336	5	.0000000000	0	.0000000000	0
722	.0000000000	0	.1582955618-	5	.0000000000	*	.0000000000	0	.1216818947	4
727	.1524555205	4	.0000000000	0	.4009420664-	3	.0000000000	0	.0000000000	0
803	.0000000000	0	.0000000000	0	.4172939739-	4	.1033342696	6	.1054048388-	6
808	.6689125411	6	.0000000000	0	.0000000000		.0000000000	0	.0000000000	0
813	.0000000000	0	.0000000000		.0000000000		.0000000000	0	.0000000000	0
818	.0000000000	0	.0000000000	*	.0000000000		.0000000000	0	.0000000000	0
823	.0000000000	0	.0000000000	0	.0000000000		.1083173175	5	.2107882001-	5
828	.0000000000	0	.0000000000	0	.9661205555-	1	.5254421003	2	.9406537104-	3
904	.1376238604	5	.2114887918-	5	.3152509181	6	.2442203826-	6	.0000000000	0
909	.1030162389	6	.0000000000	*	.0000000000		.0000000000	0	.0000000000	0
914	.0000000000	0	.0000000000	0	.0000000000		.0000000000	0	.0000000000	0
919	.0000000000	0	.0000000000	*	.0000000000		.0000000000	0	.0000000000	0
924	.0000000000	0	.0000000000	*	.0000000000		.0000000000	0	.0000000000	0
929	.0000000000	0	.0000000000	0	.0000000000		.5240639171-	5	.5430837130	4
1005	.1819402783	1-	.0000000000	*	.0000000000		.0000000000	0	.0000000000	0
1010	.2854295760	6	.1885468074-	5	.1991422306-	4	.0000000000	0	.0000000000	0
1015	.2676268826-	6	.8459686315	4	.1094800400	5	.0000000000	0	.0000000000	0
1020	.7422888090	5	.1989600764-	5	.0000000000		.0000000000	0	.2168340073	4
1025	.0000000000	0	.0000000000	*	.0000000000		.9297632177	2	.0000000000	0
1101	.0000000000	0	.0000000000	0	.3301825977	4	.1153271155-	7	.0000000000	0
1106	.0000000000	0	.0000000000	0	.0000000000		.0000000000	0	.1885468074-	5
1111	.4391970185	7	.0000000000	0	.0000000000		.0000000000	0	.1887073127	5
1116	.1917222437-	7	.0000000000	0	.0000000000		.0000000000	0	.6536977121	6
1121	.0000000000	0	.0000000000	*	.0000000000		.0000000000	0	.0000000000	0
1126	.0000000000	0	.0000000000		.0000000000		.0000000000	*	.0000000000	0
1202	.0000000000	0	.1931808165	3	.0000000000	0	.1045721812-	7	.2273862205	4
1207	.0000000000	0	.0000000000	*	.0000000000		.1991422306-	4	.0000000000	0
1212	.3342415114	7	.1273862810-	5	.1956470381	5	.1121533334	5	.0000000000	0
1217	.1576569586-	7	.1673271194	4	.0000000000	0	.1252089151-	5	.3665006317	6

"29 x 29 Matrix A"

1222	.0000000000	0	.0000000000	*	.1247106123-	5	.0000000000	0	.0000000000	0
1227	.0000000000	0	.1764121224	4	.0000000000		.0000000000	0	.0000000000	0
1303	.0000000000	0	.0000000000	0	.1425800171	4	.6737892792-	6	.1855177321	4
1308	.0000000000	0	.0000000000	0	.0000000000		.0000000000	0	.1278862810-	5
1313	.1513472040	7	.8821830093-	5	.0000000000	0	.0000000000	0	.1651194109	4
1318	.1060695183-	7	.6620139333	4	.0000000000	0	.0000000000	0	.5438769491	6
1323	.0000000000	0	.0000000000	0	.6721400032-	5	.0000000000	0	.0000000000	0
1328	.1603925408	5	.0000000000	*	.0000000000		.0000000000	0	.0000000000	0
1404	.0000000000	0	.0000000000	0	.1050351686	5	.1629515823-	6	.0000000000	0
1409	.0000000000	0	.0000000000	0	.0000000000		.1956470381	5	.8821830093-	5
1414	.2142369990	7	.0000000000	0	.0000000000		.0000000000	0	.8607917470	4
1419	.1099693602-	7	.0000000000	0	.0000000000		.0000000000	0	.4120597796	6
1424	.0000000000	0	.0000000000	0	.5206830094-	5	.0000000000	0	.0000000000	0
1429	.1043693818	5	.0000000000	0	.0000000000	0	.2399646300	5	.0000000000	0
1505	.1024655024-	0	.0000000000	0	.0000000000		.0000000000	0	.0000000000	0
1510	.2676268826-	6	.1887073127	5	.1121533334	5	.0000000000	0	.0000000000	0
1515	.1042037298	7	.2866537811-	6	.1427618785	5	.9775445338-	4	.8968755097	4
1520	.2533990430-	6	.1120507474	6	.0000000000	*	.0000000000	0	.1221170247-	5
1525	.0000000000	0	.0000000000	0	.0000000000		.5236259731-	3	.0000000000	0
1601	.0000000000	0	.0000000000	0	.0000000000		.3481976210	6	.0000000000	0
1606	.0000000000	0	.0000000000	0	.0000000000		.0000000000	0	.8459688315	4
1611	.1917222437-	7	.0000000000	0	.0000000000		.0000000000	0	.2866537811-	6
1616	.1988558802	7	.1112163267-	6	.3354713188	5	.3077875222-	5	.9157295648-	6
1621	.7969245389	4	.0000000000	*	.0000000000		.0000000000	0	.0000000000	0
1626	.0000000000	0	.0000000000	*	.0000000000		.0000000000	0	.0000000000	0
1702	.0000000000	0	.1062027047-	4	.0000000000		.3579432551	6	.0000000000	0
1707	.0000000000	0	.0000000000	0	.0000000000		.1094800400	5	.0000000000	0
1712	.1576569586-	7	.1651194109	4	.0000000000	0	.1427618785	5	.1112163267-	6
1717	.1639841941	7	.3579762830-	5	.3668268233	5	.7618149984	5	.1000440533-	7
1722	.2399376310	4	.0000000000	0	.6856066033	5	.0000000000	0	.0000000000	0
1727	.0000000000	0	.9698398057-	4	.0000000000		.0000000000	0	.0000000000	0
1803	.0000000000	0	.0000000000	0	.0000000000		.3313806204	6	.0000000000	0
1808	.0000000000	0	.0000000000	0	.0000000000		.0000000000	0	.1673271194	4
1813	.1060695183-	7	.8607917470	4	.9775445338-	4	.3354713188	5	.3579762830-	5

"29 x 29 Matrix A"

1818	.1494353793	7	.7820299122-	5	.0000000000	0	.2472433668	4	.1198305783-	7
1823	.7472618142	4	.0000000000	0	.3140563591	6	.0000000000	0	.0000000000	0
1828	.7494316235-	5	.0000000000	*	.0000000000		.0000000000	0	.0000000000	0
1904	.0000000000	0	.0000000000	0	.0000000000		.5834576336	5	.0000000000	0
1909	.0000000000	0	.0000000000	*	.0000000000		.0000000000	0	.6620139333	4
1914	.1099693602-	7	.8968755097	4	.3077875221-	5	.3668268233	5	.7820299122-	5
1919	.1407180846	7	.0000000000	0	.0000000000	*	.1117285796	5	.8141394397-	6
1924	.0000000000	0	.0000000000	0	.2087724571	6	.0000000000	0	.0000000000	0
1929	.4184782667-	5	.0000000000	0	.0000000000		.7200680527-	4	.8001611163-	5
2005	.1143933398	0	.0000000000	*	.0000000000		.0000000000	0	.0000000000	0
2010	.7422888090	5	.6536977121	6	.1252089151-	5	.0000000000	0	.0000000000	0
2015	.2533990430-	6	.9157295648-	6	.7618149984	5	.0000000000	0	.0000000000	0
2020	.7922590978	6	.1712443284-	6	.2251419427	5	.1407101198-	5	.3153213201	5
2025	.0000000000	0	.0000000000	*	.0000000000		.6809209483-	3	.0000000000	0
2101	.0000000000	0	.0000000000	0	.1930041152	4	.0000000000	0	.3348422218-	1
2106	.0000000000	0	.0000000000	0	.0000000000		.0000000000	0	.1989600764-	5
2111	.0000000000	0	.3665006317	6	.0000000000		.0000000000	0	.1120507474	6
2116	.7969245389	4	.1000440533-	7	.2472433668	4	.0000000000	0	.1712443284-	6
2121	.9921300572	6	.3597784223-	5	.2599568678	5	.1275801501-	6	.2162222531	4
2126	.0000000000	0	.0000000000	*	.1762507593	5	.0000000000	0	.0000000000	0
2202	.0000000000	0	.0000000000	*	.0000000000		.0000000000	0	.1196194835-	6
2207	.0000000000	0	.0000000000	*	.0000000000		.0000000000	0	.0000000000	0
2212	.0000000000	0	.5438769491	6	.0000000000		.0000000000	0	.0000000000	0
2217	.2399376310	4	.1198305783-	7	.1117285796	5	.2251419427	5	.3597784223-	5
2222	.1397444025	7	.6496881001-	5	.3036412342	4	.6237452968-	6	.4528849346	4
2227	.0000000000	0	.2077971907	6	.0000000000		.0000000000	0	.0000000000	0
2303	.0000000000	0	.0000000000	*	.0000000000		.0000000000	0	.1582955618-	5
2308	.0000000000	0	.0000000000		.0000000000		.0000000000	0	.0000000000	0
2313	.0000000000	0	.4120597796	6	.0000000000		.0000000000	0	.0000000000	0
2318	.7472618142	4	.8141394397-	6	.1407101198-	5	.2599568678	-5	.6496881001-	5
2323	.7628170177	6	.0000000000	0	.6301189950	4	.3133104375-	6	.0000000000	0
2328	.0000000000	0	.8380147004	5	.0000000000		.0000000000	0	.2103429817-	3
2404	.0000000000	0	.1139380806	0	.0000000000		.0000000000	0	.0000000000	0
2409	.0000000000	0	.2168340073	4	.0000000000		.1247106123-	5	.0000000000	0

PX 71900-9-(164)

"29 x 29 Matrix A"

2414	.0000000000	0	.1221170247-	5	.0000000000		.6856066033	5	.0000000000	0
2419	.0000000000	0	.3153213201	5	.1275801501-	6	.3036412342	4	.0000000000	0
2424	.1351326026	6	.6641036020-	5	.5339020821	5	.1532483536-	5	.2144214568-	5
2429	.0000000000	0	.0000000000	*	.0000000000		.0000000000	0	.0000000000	0
2505	.0000000000	0	.1478294679	5	.0000000000		.0000000000	0	.0000000000	0
2510	.0000000000	0	.0000000000	0	.0000000000		.6721400032-	5	.0000000000	0
2515	.0000000000	0	.0000000000	0	.0000000000		.3140563591	6	.0000000000	0
2520	.0000000000	0	.2162222531	4	.6237452971-	6	.6301189950	4	.6641036020-	5
2525	.8684594515	6	.2091887123-	6	.1293433351	6	.4142767648-	6	.8081281290	4
2601	.0000000000	0	.0000000000		.0000000000		.0000000000	0	.0000000000	0
2606	.0000000000	0	.1216818947	4	.1083173175	5	.0000000000	0	.0000000000	0
2611	.0000000000	0	.0000000000	*	.0000000000		.5206830094-	5	.0000000000	0
2616	.0000000000	0	.0000000000	0	.0000000000		.2087724571	6	.0000000000	0
2621	.0000000000	0	.4528849346	4	.3133104375-	6	.5339020821	5	.2091887123-	6
2626	.6770817700	6	.3261077612-	6	.1244318960	5	.1332447094-	6	.0000000000	0
2702	.0000000000	0	.0000000000	*	.0000000000		.0000000000	0	.0000000000	0
2707	.1524555205	4	.2107882001-	5	.0000000000		.0000000000	0	.0000000000	0
2712	.0000000000	0	.0000000000		.0000000000		.0000000000	0	.0000000000	0
2717	.0000000000	0	.0000000000	*	.0000000000		.0000000000	0	.0000000000	0
2722	.0000000000	0	.0000000000	0	.1532483536-	5	.1293433351	6	.3261077612-	6
2727	.3498604061	6	.0000000000	0	.0000000000		.0000000000	0	.0000000000	0
2803	.9019303291-	1	.0000000000	0	.1611736022-	1-	.3527649574-	4	.0000000000	0
2808	.0000000000	0	.0000000000	0	.9297632177	2	.0000000000	0	.1764121224	4
2813	.1603925408	5	.0000000000	0	.5236259731-	3	.0000000000	0	.9698398057-	4
2818	.7494316235-	5	.0000000000	0	.6809209483-	3	.1762507593	5	.2077971907	6
2823	.0000000000	0	.2144214568-	5	.4142767648-	6	.1244318960	5	.0000000000	0
2828	.3241272409	6	.1202951677-	5	.0000000000		.0000000000	0	.0000000000	0
2904	.0000000000	0	.0000000000	*	.0000000000		.4009420664-	3	.0000000000	0
2909	.0000000000	0	.0000000000	*	.0000000000		.0000000000	0	.0000000000	0
2914	.1043693818	5	.0000000000	0	.0000000000		.0000000000	0	.0000000000	0
2919	.4184782667-	5	.0000000000	0	.0000000000		.0000000000	0	.8380147004	5
2924	.0000000000	0	.8081281290	4	.1332447094-	6	.0000000000	0	.1202951677-	5
2929	.7014090267	5	.1221170247-	5	.0000000000		.6856066033	5	.0000000000	0

PA (MOT)-4-0041 (4-54)

"29 x 29 Matrix A dehydrated"

000505 41452 "Header card for input"

1	.3139500675	4	.5048596477-	4	.6178035335	4	.6385646884-	4	.1515033341	4
2	.4309664059-	3	.1144258462	3	.0000000000	*	.9661205555-	1	.0000000000	19
3	.5048596477-	4	.1135023727	5	.2435556348-	5	.3472949305	5	.8239782252-	4
4	.2343888578	4	.6223256157-	3	.0000000000		.5254421003	2	.0000000000	19
5	.6178035335	4	.2435556348-	5	.1420318524	6	.3699632856-	6	.1475097189	6
6	.4196061725-	5	.1114095918	5	.0000000000	0	.9406537104-	3	.5240639171-	5
7	.3301825977	4	.1931808165	3	.0000000000	1	.2399646300	5	.0000000000	0
8	.1062027047-	4	.0000000000	1	.7200680527-	4	.1930041152	4	.0000000000	1
9	.2103429817-	3	.0000000000	2	.9019303291-	1	.0000000000	0	.6385646884-	4
10	.3472949305	5	.3699632856-	6	.2012897506	7	.9757463285-	6	.6139115888	6
11	.1629996030-	6	.0000000000	0	.1376238604	5	.5430837130	4	.1153271155-	7
12	.0000000000	3	.3481976210	6	.0000000000	2	.8001611163-	5	.0000000000	8
13	.1515033341	4	.8239782252-	4	.1475097189	6	.9757463285-	6	.1087084987	7
14	.7908157834-	6	.2528029819	6	.4172939739-	4	.2114887918-	5	.1819402783	1
15	.0000000000	0	.1045721812-	7	.1425800171	4	.0000000000	0	.1024655024-	0
16	.0000000000	0	.3579432551	6	.0000000000	1	.1143933398	0	.3348422219-	1
17	.0000000000	1	.1139380806	0	.0000000000	2	.1611736022-	1	.0000000000	0
18	.4309664059-	3	.2343888578	4	.4196061725-	5	.6139115888	6	.7908157836-	6
19	.2930202641	5	.1330461241-	7	.1033342696	6	.3152509181	6	.0000000000	1
20	.2273862205	4	.6737892793-	6	.1050351686	5	.0000000000	2	.3313806204	6
21	.0000000000	2	.1196194835-	6	.0000000000	1	.1478294679	5	.0000000000	1
22	.3527649574-	4	.0000000000	0	.1144258462	3	.6223256157-	3	.1114095918	5
23	.1629996030-	6	.2528029819	6	.1330461241-	7	.8233909877	6	.1054048388-	6
24	.2442203826-	6	.0000000000	2	.1855177321	4	.1629515823-	6	.0000000000	3
25	.5834576336	5	.0000000000	2	.1582955618-	5	.0000000000	1	.1216818947	4
26	.1524355205	4	.0000000000	0	.4009420664-	3	.0000000000	3	.4172939739-	4
27	.1033342696	6	.1054048388-	6	.6689125411	6	.0000000000	16	.1083173175	5
28	.2107882001-	5	.0000000000	1	.9661205555-	1	.5254421003	2	.9406537104-	3
29	.1376238604	5	.2114887918-	5	.3152509181	6	.2442203826-	6	.0000000000	0
30	.1030162389	6	.0000000000	21	.5240639171-	5	.5430837130	4	.1819402783	1
31	.0000000000	3	.2854295760	6	.1885468074-	5	.1991422306-	4	.0000000000	1
32	.2676268826-	6	.8459686315	4	.1094800400	5	.0000000000	1	.7422888090	5
33	.1989600764-	5	.0000000000	1	.2168340073	4	.0000000000	2	.9297632177	2
34	.0000000000	2	.3301825977	4	.1153271155-	7	.0000000000	4	.1885468074-	5

PX 71900-9-(164)

"29 x 29 Matrix A dehydrated"

35	.4391970185	7	.0000000000	2	.1887073127	5	.1917222437-	7	.0000000000	2
36	.6536977121	6	.0000000000	10	.1931808165	3	.0000000000	0	.1045721812-	7
37	.2273862205	4	.0000000000	2	.1991422306-	4	.0000000000	0	.3342415114	7
38	.1278862810-	5	.1956470381	5	.1121533334	5	.0000000000	0	.1576569586-	7
39	.1673271194	4	.0000000000	0	.1252089151-	5	.3665006317	6	.0000000000	1
40	.1247106123-	5	.0000000000	2	.1764121224	4	.0000000000	4	.1425800171	4
41	.6737892792-	6	.1855177321	4	.0000000000	3	.1278862810-	5	.1513472040	7
42	.8821830093-	5	.0000000000	1	.1651194109	4	.1060695183-	7	.6620139333	4
43	.0000000000	1	.5438769491	6	.0000000000	1	.6721400032-	5	.0000000000	1
44	.1603925408	5	.0000000000	5	.1050351686	5	.1629515823-	6	.0000000000	3
45	.1956470381	5	.8821830093-	5	.2142369990	7	.0000000000	2	.8607917470	4
46	.1099693602-	7	.0000000000	2	.4120597796	6	.0000000000	1	.5206830094-	5
47	.0000000000	1	.1043693818	5	.0000000000	1	.2399646300	5	.0000000000	0
48	.1024655024-	0	.0000000000	3	.2676268826-	6	.1887073127	5	.1121533334	5
49	.0000000000	1	.1042037298	7	.2866537811-	6	.1427618785	5	.9775445338-	4
50	.8968755097	4	.2533990430-	6	.1120507474	6	.0000000000	1	.1221170247-	5
51	.0000000000	2	.5236259731-	3	.0000000000	3	.3481976210	6	.0000000000	4
52	.8459686315	4	.1917222437-	7	.0000000000	2	.2866537811-	6	.1988559802	7
53	.1112163267-	6	.3354713188	5	.3077875222-	5	.9157295648-	6	.7969245389	4
54	.0000000000	7	.0000000000	1	.1062027047-	4	.0000000000	0	.3579432551	6
55	.0000000000	3	.1094800400	5	.0000000000	*	.1576569586-	7	.1651194109	4
56	.0000000000	0	.1427618785	5	.1112163267-	6	.1639841941	7	.3579762830-	5
57	.3668268233	5	.7618149984	5	.1000440533-	7	.2399376310	4	.0000000000	0
58	.6856066033	5	.0000000000	2	.9698398057-	4	.0000000000	5	.3313806204	6
59	.0000000000	4	.1673271194	4	.1060695183-	7	.8607917470	4	.9775445338-	4
60	.3354713188	5	.3579762830-	5	.1494353793	7	.7820299122-	5	.0000000000	0
61	.2472433668	4	.1198305783-	7	.7472618142	4	.0000000000	0	.3140563591	6
62	.0000000000	1	.7494316235-	5	.0000000000	6	.5834576336	5	.0000000000	4
63	.6620139333	4	.1099693602-	7	.8968755097	4	.3077875221-	5	.3668268233	5
64	.7820299122-	5	.1407180846	7	.0000000000	1	.1117285796	5	.8141394397-	6
65	.0000000000	1	.2087724571	6	.0000000000	1	.4184782667-	5	.0000000000	1
66	.7200680527-	4	.8001611163-	5	.1143933398	0	.0000000000	3	.7422888090	5
67	.6536977121	6	.1252089151-	5	.0000000000	1	.2533990430-	6	.9157295648-	6
68	.7618149984	5	.0000000000	1	.7922590978	6	.1712443284-	6	.2251419427	5

PX 71900-9(164)

"29 x 29 Matrix A dehydrated"

69	.1407101198-	5	.3153213201	5	.0000000000	2	.6809209483-	3	.0000000000	2
70	.1930041152	4	.0000000000	0	.3348422218-	1	.0000000000	3	.1989600764-	5
71	.0000000000	0	.3665006317	6	.0000000000	1	.1120507474	6	.7969245389	4
72	.1000440533-	7	.2472433668	4	.0000000000	0	.1712443284-	6	.9921300572	6
73	.3597784223-	5	.2599568678	5	.1275801501-	6	.2162222531	4	.0000000000	1
74	.1762507593	5	.0000000000	5	.1196194835-	6	.0000000000	5	.5438769491	6
75	.0000000000	2	.2399376310	4	.1198305783-	7	.1117285796	5	.2251419427	5
76	.3597784223-	5	.1397444025	7	.6496881001-	5	.3036412342	4	.6237452968-	6
77	.4528849346	4	.0000000000	0	.2077971907	6	.0000000000	6	.1582955618-	5
78	.0000000000	5	.4120597796	6	.0000000000	2	.7472618142	4	.8141394397-	6
79	.1407101198-	5	.2599568678	5	.6496881001-	5	.7628170177	6	.0000000000	0
80	.6301189950	4	.3133104375-	6	.0000000000	1	.8380147004	5	.0000000000	1
81	.2103429817-	3	.0000000000	0	.1139380806	0	.0000000000	3	.2168340073	4
82	.0000000000	0	.1247106123-	5	.0000000000	1	.1221170247-	5	.0000000000	0
83	.6856066033	5	.0000000000	1	.3153213201	5	.1275801501-	6	.3036412342	4
84	.0000000000	0	.1351326026	6	.6641036020-	5	.5339020821	5	.1532483536-	5
85	.2144214568-	5	.0000000000	5	.1478294679	5	.0000000000	5	.6721400032-	5
86	.0000000000	3	.3140563591	6	.0000000000	1	.2162222531	4	.6237452971-	6
87	.6301189950	4	.6641036020-	5	.8684594515	6	.2091887123-	6	.1293433351	6
88	.4142767648-	6	.8081281290	4	.0000000000	5	.1216818947	4	.1083173175	5
89	.0000000000	4	.5206830094-	5	.0000000000	3	.2087724571	6	.0000000000	1
90	.4528849346	4	.3133104375-	6	.5339020821	5	.2091887123-	6	.6770817700	6
91	.3261077612-	6	.1244318960	5	.1332447094-	6	.0000000000	5	.1524555205	4
92	.2107882001-	5	.0000000000	14	.1532483536-	5	.1293433351	6	.3261077612-	6
93	.3498604061	6	.0000000000	3	.9019303291-	1	.0000000000	0	.1611736022-	1
94	.3527649574-	4	.0000000000	2	.9297632177	2	.0000000000	0	.1764121224	4
95	.1603925408	5	.0000000000	0	.5236259731-	3	.0000000000	0	.9698398057-	4
96	.7494316235-	5	.0000000000	0	.6809209483-	3	.1762507593	5	.2077971907	6
97	.0000000000	0	.2144214568-	5	.4142767648-	6	.1244318960	5	.0000000000	0
98	.3241272409	6	.1202951677-	5	.0000000000	5	.4009420664-	3	.0000000000	5
99	.1043693818	5	.0000000000	3	.4184782667-	5	.0000000000	2	.8380147004	5
100	.0000000000	0	.8081281290	4	.1332447094-	6	.0000000000	0	.1202951677-	5
101	.7014090267	5	.3153213201	5	.0000000000	2	.6809209483-	3	.0000000000	2

PX 71900-9-(164)

"29 x 29 Matrix A - Largest 27 Eigenvalues"

1	.6050891256	7	.4820657418	7	.3700033876	7	.3225412389	7	.2877327277	7
2	.1646348332	7	.1490469395	7	.1347716962	7	.1225370950	7	.1129763458	7
3	.1094844363	7	.7922861997	6	.6464302894	6	.5105663229	6	.2580032290	6
4	.2201522604	6	.1811715067	6	.1406394162	6	.1343232215	6	.9338440632	5
5	.5232201519	5	.3466753013	5	.2798447725	5	.1743699669	5	.9583780118	4
6	.2743508278	4	.1721665082	4	.0000000000		.0000000000	0	.0000000000	0

"Eigenvectors of last 11 Eigenvalues obtained"

101	.1697784185-	1-	.6755937912	1-	.3457751454-	0	.1814290725	0	.4058064022	0
106	.2874679484-	1-	.1092355938-	0	.1728186382-	1-	.1517315504	0	.1450655492	0
111	.9123459799	1-	.4039786652	1-	.2948166163	1-	.5848954248	1-	.1642059764	0
116	.1244212824	0	.2480522250-	0	.6521085402	1-	.1602914272	0	.9849787016	1-
121	.2427652997-	0	.1974710336	1-	.8591464683	1-	.4886983299	0	.1219075436-	0
126	.1490918836-	0	.1515251822-	0	.2346617991-	0	.2057953153	0	.2124168233-	1-
02	.8314540038	1-	.3201122824-	0	.1774026901	0	.4023391958	0	.2868576789-	1-
207	.9138417021-	1-	.1829969611-	1-	.1996857545	0	.1821323809-	0	.6357639569	1-
212	.1717095256	0	.2659664187	2-	.3263859225-	1-	.2572519443-	1-	.4394512906	1-
217	.6806955943	1-	.6459460649	1-	.7992108421-	1-	.2550118370	1-	.7852701205-	1-
222	.9677660624	1-	.1073834896-	0	.6037754732-	0	.1782293867	0	.6350016265-	1-
227	.2545626737-	0	.2624732735	0	.1114319574	1-	.1325396579-	1-	.5334475993	1-
303	.1620893973-	0	.1435047883	0	.2972567397	0	.1179961371-	1-	.5001866488-	1-
308	.6481718209	2-	.1386099989	0	.2881271292-	0	.1548336312	1-	.1541665227	0
313	.1525161930-	0	.1609666900-	1-	.1834046828-	0	.1129682931-	0	.1365687921	0
318	.3122803859-	0	.6186870996	1-	.1811915179-	0	.8749586086	1-	.2438235325-	0
323	.2313591327	0	.1955914483	0	.3511148665-	1-	.2620749084	0	.4324840206	0
328	.6192496442	1-	.3006676827-	0	.9607619630	4-	.3521373071	2-	.4685902755	1-
404	.6393032289	1-	.9352180874	1-	.1564768509-	1-	.2182667306-	1-	.5753742916	4-
409	.7728232204	1-	.2161610841-	0	.6024493102-	1-	.1114373095	0	.1370456117	0
414	.2412643761-	1-	.2944370100-	0	.3231233726-	0	.2058412532	0	.2996875851	0
419	.1183740303-	0	.4367102724-	0	.1664313204	0	.1957903960	0	.2054768967-	0
424	.7055714447	1-	.1829903787-	0	.1071008541-	0	.3954392783-	1-	.4003045901-	0
429	.2193589783	0	.2078166822-	1-	.5997832757	1-	.1686872136-	0	.6101353041-	1-
505	.4541021708-	1-	.7863391329-	2-	.1349249255-	1-	.4557548178	2-	.2167726330-	1-
510	.5546211914	2-	.2984743251	1-	.5053491244	1-	.1222528709	0	.6495213538	1-
515	.1146143609	0	.2057434154	0	.1740178109	0	.2517806679	0	.2282384439	0

PX 71900-9-(164)

520	.2953384703	0	.1619788842	0	.1447275071	0	.3142234047	0	.3534751444-	0
525	.2472226701-	0	.1401179015	0	.2432285432	0	.4693848755-	0	.1371039500-	0
601	.9872538702	1-	.2602498939-	0	.4556905858	0	.1924160211	0	.1337630529	0
606	.1374329011-	1-	.5344956371	1-	.1107903066	1-	.2635041571	0	.1116530260	0
611	.2908455711	1-	.4470656228-	1-	.8331004188	1-	.6903861859	1-	.3254714380-	1-
616	.1198812814-	0	.2463240048-	0	.2197142242	0	.2178939709	0	.2041412889-	0
621	.2853843273-	0	.2024274199	0	.2767826831	0	.1051527654-	0	.8090153012	1-
626	.1931608232	1-	.1784416024-	1-	.7835559702	2-	.3597428473-	0	.1019303177	0
702	.2561387440-	0	.3238764402	0	.1390626257	0	.7098771774	1-	.1460346445-	1-
707	.1583348658	0	.3137682609	1-	.5754754076	0	.1152095186	0	.8347188007	1-
712	.8227234336	1-	.7917191980-	1-	.2729067231-	1-	.1008935401	0	.1765852304	0
717	.1601919207	0	.1914955548-	0	.1272063791-	0	.2047718818	0	.1495487239	0
722	.1918084555-	0	.1560439940-	0	.7616729855-	1-	.1409317877-	0	.4474957896	1-
727	.9964869954	1-	.1043866043-	0	.3308523595	0	.3120370141-	1-	.6010294681	1-
803	.6141797702-	1-	.5803298607-	1-	.9705684006-	1-	.8509412051-	2-	.1116441731	0
808	.1425485252	1-	.3345808888	0	.2015108966-	1-	.9353284944	2-	.8510108628	1-
813	.6837600983	1-	.2418526175	1-	.2472209480	1-	.1103324698	0	.3293018632	0
818	.2061056146	0	.3541688532	1-	.1576327795	0	.3847625000	0	.2297059867	0
823	.1057026602	1-	.3945798687	0	.2271121982	0	.1507579325-	0	.2176791400-	0
828	.2055625735	0	.3616729594-	0	.4028024152-	0	.8369742701	0	.2309429004	1-
904	.1241868296-	0	.2200037762-	0	.1922119896-	2-	.1862118391	0	.2949753932	1-
909	.4615453073	0	.7436527908-	3-	.5819631179-	1-	.1389994164-	0	.1027053755-	1-
914	.2343719156	1-	.2749795002-	1-	.9006965406-	1-	.1836035234-	0	.2913061208-	1-
919	.2578171728	1-	.9245449595-	1-	.1601742298-	0	.3353042183-	1-	.2926734688	1-
924	.8867986790-	1-	.3590653941-	1-	.3372141267	1-	.4296493994	1-	.3331653875-	1-
929	.4687968794	1-	.1918261949-	1-	.1337762857	1-	.2127323634	1-	.1208657289	1-
1005	.1402282325	1-	.1127049421-	2-	.8704605938-	2-	.5376445899	3-	.1616780489-	1-
1010	.1931098290-	3-	.2492475922	2-	.2137876390	1-	.7163426069	1-	.2721548472	1-
1015	.8279944112-	2-	.5123606982-	2-	.4697118496	1-	.2279688404	0	.1262890288	0
1020	.1031313519-	1-	.4919491178	1-	.2789634266	0	.2374814917	0	.1027065596	0
1025	.3272412981	0	.3983715423	0	.2569282457	0	.3027827199	0	.5812507830	0
1101	.6548873728-	0	.1322672434	0	.4436966511	0	.2430701760	0	.2773814189	0
1106	.4233536414-	3-	.1411090518-	0	.2084736623-	1-	.3100173512-	0	.1090777226	0
1111	.9652626801	1-	.1440845923	0	.7640430686-	3-	.2027677427-	1-	.5097229135	1-

1116	.1041514800	0	.1455984896	0	.1542009111-	2-	.2660670653-	1-	.8545663034	1-
1121	.1083388567	0	.1924152284-	2-	.3142390786-	1-	.3320401456	1-	.2622282131-	2-
1126	.3080861266-	1-	.2706741637-	1-	.2473936938-	2-	.3564298578-	1-	.3254714380-	1-

"29 x 29 Matrix A – Ay and Ay - λy for 5 Last Eigenvalues and Eigenvectors Obtained"

Ay

1	.2852466907	4	.7167909440-	4	.9063514133	4	.3891594207	4	.1986554283	4
2	.4086722617-	3	.4430918353	4	.8780640983	3	.1610437891	5	.3224078411	4
3	.2335918251	4	.2302347264	4	.2215584315-	4	.7637150934-	3	.2823453282	4
4	.4941644253	4	.4482887045	4	.5358902828-	4	.3559803496-	4	.5730433769	4
5	.4185042341	4	.5367659211-	4	.4366808899-	4	.2131501583-	4	.3943901951-	4
6	.1252293543	4	.2788617120	4	.2921205043-	4	.9258729738	4	.0000000000	0

Ay - λy

1	.2504587173	3-	.5848407745-	3-	.1263141632	2-	.6785392761-	3-	.1112222671	3-
2	.1943171024-	2-	.9536743164-	4-	.2238154411	4-	.4587173462	3-	.2590417862	3-
3	.1322746277	2-	.1255750656-	2-	.4724264145	3-	.1045763492	3-	.3050565719	3-
4	.1109361648-	2-	.1149177551-	3-	.1678466797	3-	.5263090133	3-	.2999305725-	3-
5	.5199909210-	3-	.1478195190	3-	.7002353668	3-	.4503726959	3-	.4549026489	3-
6	.3117322922-	4-	.3548860550	3-	.4901885986-	3-	.5884170532-	3-	.0000000000	0

1	.5440983961-	3	.1048014229	4	.1070945040-	4	.1011921139-	4	.1692379541-	4
2	.1483782086-	3	.1946738062	4	.2485619473	3	.5834085742	4	.3513752676-	3
3	.1630930404	3	.1483907992	4	.1192272509	4	.4217181994	3	.4310814497	3
4	.1923865984	4	.5742036509	4	.3593862558	4	.6175642175	3	.2748642092	4
5	.6709101952	4	.4005382667	4	.1843134222	3	.6880287668	4	.3960154893	4
6	.2628765960-	4	.3795670600-	4	.3584393749	4	.6306490416-	4	.0000000000	0

PX 71900-9-(164)

1	.4420876503	3-.6553530693-	3-.2127885818	4-.1532733440-	3-.2567768097	3-
2	.3810673952	3-.1014292240-	2-.1311376691	3-.1082420349-	3-.7841587066-	3-
3	.1581385731-	3-.6322860718	3-.2524852752	3-.1296848058-	3-.2364650368	2-
4	.9267926216-	3-.1010894775	2-.3602504730-	3-.1056790352	3-.1633167267-	3-
5	.4863739013-	3-.1376867294	3-.2712234854-	3-.1966953277-	3-.2461671829	3-
6	.3905296325-	3-.1571178436-	3-.1652240753-	3-.2198219299-	3-.0000000000	0

1	.3860370252-	4	.6104620982	4	.2213303524	3	.1190178442-	4	.2108467695-	4
2	.1842129259-	2	.1784613621	4	.2826977963	3	.4423348312	4	.7126259464-	1
3	.5577404519-	3	.1332140528-	4	.9843081920-	2	.2246167689	3	.2635366673-	3
4	.8632075419-	3	.1759616660-	4	.2791814680-	3	.2470861891	3	.8860629077-	3
5	.1535074602-	4	.3213483627-	3	.2804917842	3	.8498887871-	3	.3441209253-	3
6	.3231788901	3	.4117666688	3	.3192984558-	3	.4492850306	3	.0000000000	0

1	.4743337631-	3-.3623962402-	3-.2453178167-	3-.8261799812	3-.1211166382	3-
2	.1181596890-	3-.2995133400	3-.1343339681-	3-.4270076752-	3-.7453656290	3-
3	.2039968967	3-.6852746010-	3-.2457424998-	3-.1215934753-	3-.2360776067-	2-
4	.2179443836	3-.8629560470-	3-.8715689182-	4-.1202076673-	3-.6519556045	3-
5	.3635883331-	5-.1727342605-	3-.3279745579-	4-.4321336746-	3-.5468875170-	3-
6	.2860575914	3-.1317709684	3-.7416307926-	4-.4095435142	3-.0000000000	0

1	.5262494217-	2	.3670434084	2	.5836455391	2	.3316032743	2	.3847212355	2
2	.3092504680-	1	.2388179941-	2	.1475005865	1	.4435703933-	2	.5295424619-	0
3	.6838390588	1	.5865302533	2	.1965289407	3	.7466571380	2	.2271582305-	2
4	.1405645601-	2	.1288661291	3	.6254343042	3	.3464750232	3	.2829406340-	2
5	.1349666774	3	.7653385296	3	.6515322532	3	.2817761727	3	.8977889153	3
6	.1092935718	4	.7048847961	3	.8306868796	3	.1594666481	4	.0000000000	0

PX 71900-9-(164)

1	.2733180299	2-	.2706145867	2-	.1253914088	2-	.7146596908	3-	.3918819129	3-
2	.4352668766-	3-	.6409706548-	3-	.2651748946-	4-	.5328077823-	3-	.2559524146	3-
3	.2622641623	3-	.2096276730	3-	.2464428544-	3-	.1937970519-	3-	.2701580524	3-
4	.2021472901	3-	.2944022417	3-	.9647011757-	4-	.2738833427	4-	.1083612442	3-
5	.2973526716	4-	.5951523781	4-	.1850426197-	3-	.1238435507-	3-	.2950429916-	3-
6	.9459257126	4-	.2720952034	4-	.1901388168-	4-	.1463890075	3-	.0000000000	0

1	.1127496843-	4	.2277200880	3	.7638971079	3	.4184852155	3	.4775577502	3
2	.7289659832-	0	.2429425149-	3	.3589217010-	2	.5337460612-	3	.1877951176	3
3	.1661858749	3	.2480653283	3	.1315410038-	1	.3490980232-	2	.8775775775	2
4	.1793139050	3	.2506719238	3	.2654883854-	1	.4580782139-	2	.1471273158	3
5	.1865230494	3	.3312833250-	1	.5410145628-	2	.5716629106	2	.4514717266-	1
6	.5304217386-	2	.4660101890-	2	.4259395062-	1	.6136540591-	2	.0000000000	0

1	.1210570335-	3-	.1935139298	3-	.7665157318	4-	.2190023660-	3-	.1531392336-	3-
2	.9280181257-	4-	.1234561205	4-	.1236610114	4-	.1299381256-	4-	.1885816455-	3-
3	.3013759851-	4-	.8308142423-	4-	.1623353455	4-	.1191534102	4-	.5436092615	3-
4	.6134063005-	4-	.8834898472	4-	.6061152089-	4-	.1616589725	4-	.3806129098-	3-
5	.1771226525-	3-	.8745200466-	4-	.1138821244-	4-	.9864196181	4-	.2568820491-	4-
6	.6124749779-	4-	.6714835763	5-	.1042240765-	3-	.1218877733-	3-	.0000000000	0

Code

40000	00000	45	00000	41076	JUMP TO STARTER
40001	00001	00	00000	00000	2N
40002	00002	00	00000	00000	N-1
40003	00003	00	00000	00000	$\overline{E}$
40004	00004	00	00000	00000	E
40005	00005	00	00000	00000	N
40006	00006	00	00000	00000	$2NX2^{15}$
40007	00007	00	00007	77777	LAST ITERATION SWITCH
40010	00010	00	00000	00000	INDEX
40011	00011	00	00000	00000	AX
40012	00012	00	00000	00000	RQ
40013	00013	00	00000	00000	TEMPORARY
40014	00014	00	00000	00000	CELLS
40015	00015	00	41452	00000	L (MATRIX AMD) $X2^{15}$
40016	00016	00	41452	41452	L (AMD) $X2^{15}$ + L (AMD)
40017	00017	00	00000	00000	α
40020	00020	00	00000	00000	
40021	00021	00	00420	00420	L (V_1) $X2^{15}$ + L (V_1)
40022	00022	00	00420	00420	L (V_2) $X2^{15}$ + L (V_2)
40023	00023	00	00420	00420	L (A) $X2^{15}$ + L (A)
40024	00024	00	00000	00000	TYPING INDEX
40025	00025	00	00000	00000	OP 1
40026	00026	00	00000	00000	
40027	00027	00	00000	00000	OP 2
40030	00030	00	00000	00000	
40031	00031	00	00000	00000	RES
40032	00032	00	00000	00000	
40033	00033	00	00000	00000	INNER
40034	00034	00	00000	00000	PRODUCT
40035	00035	00	00000	00000	S OF V_2 + SV_1
40036	00036	00	00000	00000	
40037	00037	00	00002	00000	

PX 71900-9-(164)

```
40040  00040  00 00000 00000     ZERO
40041  00041  00 00000 00002
40042  00042  00 00000 00001
40043  00043  00 00000 00000
40044  00044  00 00000 00000     TEST INDEX
40045  00045  00 00000 00000          INDEX
40046  00046  00 00000 00000          INDEX
40047  00047  00 00000 00000     ITERATION COUNT
40050  00050  37 76000 76002     ARITHMETIC PACKAGE CA 001
40051  00051  45 00000 30000
40052  00052  45 00000 00076
40053  00053  45 00000 00142
40054  00054  11 00027 20000
40055  00055  47 00056 00050
40056  00056  11 00025 20000
40057  00057  47 00063 00060
40060  00060  11 00040 00031
40061  00061  11 00040 00032
40062  00062  45 00000 00051
40063  00063  54 20000 00042
40064  00064  73 00027 10000
40065  00065  11 00040 00025
40066  00066  74 10000 00025
40067  00067  11 20000 00031
40070  00070  23 00026 00030
40071  00071  11 00025 20000
40072  00072  47 00074 00073
40073  00073  21 00026 00042
40074  00074  11 00026 00032
40075  00075  45 00000 00051
40076  00076  11 00025 20000
40077  00077  47 00100 00120
```

```
40100  00100  11 00027 20000
40101  00101  47 00105 00102
40102  00102  11 00025 00031
40103  00103  11 00026 00032
40104  00104  45 00000 00051
40105  00105  11 00026 20000
40106  00106  36 00030 00032
40107  00107  46 00116 00110
40110  00110  11 00025 20000
40111  00111  11 00027 00025
40112  00112  11 20000 00027
40113  00113  11 00026 20000
40114  00114  11 00030 00026
40115  00115  11 20000 00030
40116  00116  12 00032 20000
40117  00117  42 00141 00123
40120  00120  11 00027 00031
40121  00121  11 00030 00032
40122  00122  45 00000 00051
40123  00123  16 20000 00124
40124  00124  54 00027 00000
40125  00125  35 00025 20000
40126  00126  11 00040 00032
40127  00127  74 20000 00032
40130  00130  11 20000 00031
40131  00131  47 00133 00132
40132  00132  13 00032 00026
40133  00133  11 00032 20000
40134  00134  42 00140 00136
40135  00135  23 00026 00207  (0026)-110
40136  00136  21 00032 00026
40137  00137  45 00000 00051
```

PX 71900-9-(164)

```
40140  00140  00 00000 00046
40141  00141  00 00000 00043
40142  00142  71 00025 00027
40143  00143  11 00040 00025
40144  00144  74 20000 00025
40145  00145  11 20000 00031
40146  00146  47 00151 00147
40147  00147  11 00040 00032
40150  00150  45 00000 00051
40151  00151  23 00025 00141
40152  00152  21 00026 00030
40153  00153  35 00025 00032
40154  00154  45 00000 00051
40155  00155  15 00022 00165     INNER PRODUCT SET L (V2)
40156  00156  15 00021 00163     SET L  (V1)
40157  00157  11 00002 00010     SET INDEX = N-1
40160  00160  23 00033 20000     STORE
40161  00161  23 00034 20000         0
40162  00162  75 30002 00164     V1
40163  00163  11 30000 00025     ⟶ OP 1
40164  00164  75 30002 00166     V2
40165  00165  11 30000 00027     ⟶ OP 2
40166  00166  37 00051 00053     MULTIPLY
40167  00167  75 30004 00171     RES ⟶ OP 1
40170  00170  11 00031 00025     INNER PRODUCT ⟶ OP 2
40171  00171  37 00051 00052     ADD
40172  00172  11 00031 00033     STORE
40173  00173  11 00032 00034       INNER PRODUCT
40174  00174  21 00163 00037     SET FOR
40175  00175  21 00165 00037       NEXT ELEMENT
40176  00176  41 00010 00162     INNER PRODUCT COMPLETE
40177  00177  45 00000 30000     EXIT
```

PX 71900-9-(164)

40200	00200	00 00000 00000		ORTHOGONALIZATION INDEX
40201	00201	41 00200 00210		EV PART II. ORTHOGONALIZE?
40202	00202	56 30000 00203		YES: DUMMY
40203	00203	11 00004 20000		E $\longrightarrow$ (A)
40204	00204	47 00205 00210		E = 0?
40205	00205	75 30210 00211		NO: ORTHOG S.R. $\longrightarrow$ ES
40206	00206	11 40740 00210		JUMP TO ORTHOGONALIZE
40207	00207	00 00000 00110		ARITHMETIC CONSTANT
40210	00210	75 30000 00212		X
40211	00211	11 30000 30000		$\longrightarrow$ VMD
40212	00212	37 00324 00306		R.Q. S.R.
40213	00213	13 00033 00035		STORE
40214	00214	11 00034 00036		$-\mu(X) = S$
40215	00215	37 00305 00260		$V_2 + SV_1$ S.R. $AX-\mu X = \xi \longrightarrow V_1$
40216	00216	21 00047 00042		ITERATION COUNT + 1
40217	00217	41 00044 00222		READY TO TEST?
40220	00220	75 30160 01640		YES: PART I $\longrightarrow$ ES
40221	00221	11 40560 01620		JUMP TO TEST
40222	00222	56 30000 00223		NO: DUMMY
40223	00223	41 00045 00250		NEW α?
40224	00224	11 00247 00045		YES: RESET α INDEX
40225	00225	37 00324 00306		R. Q. S.R.
40226	00226	75 30004 00230		$\mu(\xi) \longrightarrow$ OP 1
40227	00227	11 00033 00025		$-\mu(X) \longrightarrow$ OP 2
40230	00230	37 00051 00052		ADD
40231	00231	11 00031 00027		$\mu(\xi) - \mu(X)$
40232	00232	11 00032 00030		$\longrightarrow$ OP 2
40233	00233	11 30000 10000		M $(\beta_1) \longrightarrow$ (Q)
40234	00234	21 00233 00257		SET FOR β_2
40235	00235	41 00046 00240		USED ALL β's?
40236	00236	15 00304 00233		YES: RESET TO β_1
40237	00237	11 00041 00046		RESET β INDEX

PX 71900-9-(164)

40240	00240	56	30000	00241	NO : MS #3 STOP TO CHANGE β.
40241	00241	11	10000	00025	β
40242	00242	23	00026	20000	$\longrightarrow$ OP 1
40243	00243	37	00051	00054	DIVIDE
40244	00244	11	00031	00017	STORE
40245	00245	11	00032	00020	$\alpha = \beta \div [\mu(\xi) - \mu(x)]$
40246	00246	56	30000	00250	DUMMY
40247	00247	00	00000	00001	α INDEX
40250	00250	75	30000	00252	X
40251	00251	11	00000	00000	$\longrightarrow V_2$
40252	00252	12	00017	00035	$\lvert\alpha\rvert$
40253	00253	11	00020	00036	$\longrightarrow$ S
40254	00254	37	00305	00260	$V_2 + SV_1 \longrightarrow V_1$: $x + \alpha \xi = x_1 \longrightarrow V_1$
40255	00255	45	00000	00256	DUMMY
40256	00256	56	30000	00201	RETURN TO ORTHOG TEST
40257	00257	00	00001	00000	
40260	00260	11	00002	00010	$V_2 +$ SV_1 S.R. SET INDEX = N-1
40261	00261	15	00021	00265	STORE L (V_1)
40262	00262	15	00022	00274	STORE L (V_2)
40263	00263	16	00021	00277	STORE L (V_1)
40264	00264	75	30002	00266	V_{11}
40265	00265	11	30000	00025	$\longrightarrow$ OP 1
40266	00266	11	00035	00027	S
40267	00267	11	00036	00030	$\longrightarrow$ OP 2
40270	00270	37	00051	00053	MULT.
40271	00271	11	00031	00025	SV_{11}
40272	00272	11	00032	00026	$\longrightarrow$ OP 1
40273	00273	75	30002	00275	V_{21}
40274	00274	11	30000	00027	$\longrightarrow$ OP 2
40275	00275	37	00051	00052	ADD
40276	00276	75	30002	00300	$V_{21} + SV_{11}$
40277	00277	11	00031	30000	$\longrightarrow V_{11}$

PX 71900-9-(164)

```
40300  00300  21 00265 00037    SET FOR L ( V12)
40301  00301  21 00274 00037    SET FOR L ( V22)
40302  00302  21 00277 00041    SET FOR L ( V12)
40303  00303  41 00010 00264    DONE ?
40304  00304  45 00410 00305    DUMMY, L (M(β))   X2^15
40305  00305  56 30000 30000    EXIT
40306  00306  37 00326 00330    R.Q. S.R. AX ⟶ V2
40307  00307  15 00021 00165    SET FOR XX
40310  00310  37 00177 00156    INNER PRODUCT S.R.
40311  00311  11 00033 00013    STORE
40312  00312  11 00034 00014      XX
40313  00313  37 00177 00155    INNER PRODUCT
40314  00314  11 00033 00025    XAX
40315  00315  11 00034 00026          ⟶ OP 1
40316  00316  11 00013 00027    XX
40317  00317  11 00014 00030          ⟶ OP 2
40320  00320  37 00051 00054    DIVIDE
40321  00321  11 00031 00033    STORE
40322  00322  11 00032 00034         μ(X) = XAX/XX
40323  00323  45 00000 00324    DUMMY
40324  00324  56 30000 30000    EXIT
40325  00325  00 00000 00000    TEMP α INDEX
40326  00326  45 00000 30000    AX S.R. EXIT
40327  00327  45 00000 00407    JUMP TO SETUP
40330  00330  15 00015 00364    SET L ( AMD )
40331  00331  16 00022 00401    SET L (AX )
40332  00332  15 00405 00350    SET L ( A )
40333  00333  11 00002 00010    SET INDEX=N-1
40334  00334  11 00040 00012    INITIAL
40335  00335  11 00040 00013      ZERO
40336  00336  11 00040 00014        SETS
40337  00337  21 00012 00002    SET INDEX=N-1
```

PX 71900-9-(164)

```
40340  00340  43 00002 00342    WAS INDEX ORIGINALLY ZERO?
40341  00341  45 00000 00343    NO:  JUMP
40342  00342  15 00021 00346    YES: SET L(X_i)
40343  00343  11 00405 20000    CONST ⟶(A)
40344  00344  43 00350 00361    WORKING STORAGE EXHAUSTED?
40345  00345  75 30002 00347    NO:   X_i
40346  00346  11 30000 00025          ⟶ OP 1
40347  00347  75 30002 00351    A_ii
40350  00350  11 30000 00027          ⟶ OP 2
40351  00351  11 00027 20000
40352  00352  47 00365 00353    M(A_ii) = 0?
40353  00353  31 00030 00020    YES: 2(FLAG) X 2^15
40354  00354  35 00346 00346    SKIP X_i's
40355  00355  23 00012 00030    CORRECT INDEX
40356  00356  46 00357 00375    INDEX < 0 ?
40357  00357  23 00346 00006    YES: (L (X_i) -2N) X 2^15
40360  00360  45 00000 00375    JUMP TO NEXT ROW
40361  00361  15 00023 00350    YES WORKING SPACE EXHAUSTED: SET L (A)
40362  00362  21 00364 00406    SET FOR NEXT BLOCK TRANSFER
40363  00363  75 30000 00345    BLOCK OF MATRIX
40364  00364  11 30000 30000          ⟶ ES
40365  00365  37 00051 00053    MULT.
40366  00366  11 00031 00025     A_ii X_i
40367  00367  11 00032 00026          ⟶ OP 1
40370  00370  11 00013 00027    (AX)_i
40371  00371  11 00014 00030          ⟶ OP 2
40372  00372  37 00051 00052    ADD
40373  00373  11 00031 00013    STORE
40374  00374  11 00032 00014      SUM
40375  00375  21 00346 00037    SET
40376  00376  21 00350 00037      FOR NEXT ELEMENT
40377  00377  41 00012 00343      INNER PRODUCT DONE?
```

PX 71900(9)-(164)

```
40400  00400  75 30002 00402    YES:  STORE
40401  00401  11 00013 30000          (AX)1
40402  00402  21 00401 00041    SET FOR  (AX)2
40403  00403  41 00010 00335     AX COMPLETED?
40404  00404  45 00000 00326    YES:  JUMP TO EXIT
40405  00405  11 02000 00027    CONST.
40406  00406  00 00000 00000    KX2^15 = NUMBER OF ES W.S.
40407  00407  11 00040 00406    0 ⟶ TEST INDEX
40410  00410  15 00023 00406    STORE
40411  00411  11 00420 20000       K
40412  00412  36 00406 00406
40413  00413  11 00421 10000    MASK ⟶ (Q)
40414  00414  53 00406 00363    SET REPEAT
40415  00415  23 00015 00406    L  (AMD)X 2^15  -KX2^15
40416  00416  16 00023 00364    SET L  (A)
40417  00417  45 00000 00326    JUMP TO EXIT
40420  00420  00 02000 00000    CONST.                V1
40421  00421  00 07777 00000    CONST.
40422  00422  00 00000 00000
40423  00423  00 00000 00000
40424  00424  00 00000 00000
40425  00425  00 00000 00000

40426         54 01637 10000    PART I PATCH TX2^6R ⟶(A) , (Q)
40427         53 20000 40564    STORE NEW T
40430         16 40434 40427    RESTORE
40431         11 40435 40426     RESTORE
40432         45 00000 01716     RETURN TO TEST
40433         00 00000 00077    MASK
40434         00 00000 40564
40435         54 01637 10000
40436         00 00000 00000
```

PX 71900-9-(164)

```
40437          00 00000 00000
40440          75 30274 01561     PART III→ES. JUMP
40441          11 40444 01504       TO INTERMEDIATE START
40442          75 30114 01504     PART III→ES. JUMP TO
40443          11 40444 01504     PART III

40444  01504  11 00033 10000     M(ν)→(Q)
40445  01505  37 01743 01734     TYPE M(ν)
40446  01506  55 00034 10033
40447  01507  37 01743 01734     TYPE EXP(ν)
40450  01510  56 30000 01511     DUMMY
40451  01511  55 00004 10001      2E→(Q)
40452  01512  21 01515 10000     SET FOR E VALUE
40453  01513  21 01527 10000     SET FOR INNER PRODUCT
40454  01514  75 30002 01516     STORE
40455  01515  11 00033 30000       E VALUE
40456  01516  71 00001 00004      2NE→(A)
40457  01517  35 01523 01523     SET EV REGION
40460  01520  75 30000 01522       Y
40461  01521  11 30000 30000           → V1
40462  01522  75 30000 01524       Y
40463  01523  11 30000 30000        → EV REGION
40464  01524  15 00021 00165     SET FOR INNER PRODUCT
40465  01525  37 00177 00156     INNER PRODUCT
40466  01526  75 30002 01530     STORE
40467  01527  11 00033 41052           YY FOR ORTHOG
40470  01530  21 00004 00042     E + 1
40471  01531  13 00007 00007     RESET LAST ITERATION SWITCH
40472  01532  11 00004 10000     E→(Q)
40473  01533  11 00003 20000     E→(A)
40474  01534  56 20000 01535     MS #2 STOP TO CHANGE E
40475  01535  11 20000 00003      E→(003)
```

PX 71900-9-(164)

```
40476  01536  11 10000 00004    E ⟶(004)
40477  01537  42 00004 01565    E > Ē ?
40500  01540  43 00004 01570    NO E < Ē :  E = Ē ?
40501  015A1  56 30000 01542    NO:DUMMY
40502  01542  11 01774 00044    TEST INDEX = 1
40503  01543  11 00040 00047    ITERATION INDEX = 0
40504  01544  11 01621 00024    TYPE INDEX = 1
40505  01545  11 00040 00045       INDEX = 0
40506  01546  11 00040 00200    ORTHOG INDEX = 0
40507  01547  61 00000 01757    C•R•
40510  01550  61 00000 01757    C•R•
40511  01551  55 00004 10033    EX2^33 ⟶(Q)
40512  01552  37 01743 01735    TYPE E
40513  01553  71 00006 00004     2EN X 2^15 ⟶(A)
40514  01554  35 01556 01556    SET FOR X_E
40515  01555  75 30000 01613     X_E ⟶ V_1
40516  01556  11 30000 30000              JUMP TO TYPE T
40517  01557  61 00000 01757    C•R•
40520  01560  56 30000 00201    JUMP TO ORTHOG TEST
40521  01561  75 30420 01532    INTERMEDIATE START
40522  01562  11 40001 00001    RESTORE ES, PART II ⟶ES
40523  01563  11 00003 20000    Ē ⟶(A)
40524  01564  56 30000 01536    JUMP TO STORE E
40525  01565  37 76000 76001    E > Ē : ALARM
40526  01566  11 00003 20000    Ē ⟶ (A)
40527  01567  56 00000 01540    RETURN TO TEST E = Ē
40530  01570  23 10000 20000    YES E = Ē : OUTPUT STOP TO
40531  01571  56 00000 01572              SET M
40532  01572  11 10000 00010    STORE M
40533  01573  16 00003 01575    SET L .( E VALUES )
40534  01574  37 76600 76601    OUTPUT
40535  01575  00 30000 30000       E VALUES
```

PX 71900-9-(164)

```
40536  01576  13 00010 10000
40537  01577  21 10000 00003    E - M ⟶(Q)
40540  01600  71 10000 00006    (E-M)2NX2^15 ⟶(A)
40541  01601  35 01610 01610     SET L(1ST VECTOR FOR OUTPUT)
40542  01602  55 00010 10017      MX2^15
40543  01603  15 10000 01611     STORE M IN OUTPUT PARAMETER
40544  01604  11 00000 76463     SET MT S.R.
40545  01605  37 76463 76430     MT DUMP EV'S
40546  01606  11 01612 76463     RESTORE MT S.R.
40547  01607  37 76600 76603     OUTPUT
40550  01610  00 30000 00000         E VECTORS
40551  01611  00 30000 30000
40552  01612  56 00000 01532     STOP. READY TO JUMP TO MS #2
40553  01613  37 01716 01706     LOCATE T  (IN PART I)
40554  01614  55 01637 10033      TX2^33 ⟶(Q)
40555  01615  37 01743 01734      TYPE T
40556  01616  45 00000 01557     RETURN
40557  01617  00 00000 00000
40560  01620  00 00000 00000     PART I
40561  01621  00 00000 00001     TYPING INDEX
40562  01622  00 00000 00000
40563  01623  00 00000 00006
40564  01624  37 37373 74042     T
40565  01625  35 35353 63636       STORAGE
40566  01626  35 35353 53535
40567  01627  34 34343 43434
40570  01630  33 33033 33333
40571  01631  32 32323 23232
40572  01632  31 31313 13131
40573  01633  30 30303 03030
40574  01634  25 25252 52525
40575  01635  20 20202 02020
```

```
40576  01636  14 14141 41414
40577  01637  00 00000 00000    T
40600  01640  11 01774 00044    RESET TEST INDEX. S = V1 , AX = V2
40601  01641  11 30000 20000    M(S1) ⟶ (A)
40602  01642  47 01643 01650    S1 = 0 ?
40603  01643  23 30000 30000    NO: EXP(AX)1 - EXP(S) → (EXP V2)
40604  01644  21 01641 00037    SET FOR M(S2)
40605  01645  21 01643 01775    SET FOR EXP(AX)2 - EXP(S2)
40606  01646  41 01776 01641    ALL EXPONENTS STORED
40607  01647  56 30000 01655    SKIP S=0 TEST
40610  01650  55 01643 10025    YES S1 = 0 : SET
40611  01651  16 10000 01652         L(EXP V2)
40612  01652  11 01765 30000      64s ⟶ (EXP V2)
40613  01653  45 00000 01644    RETURN TO STORE EXPONENTS
40614  01654  00 00000 00000    FIND SMALLEST EXPONENT
40615  01655  21 01661 00037    SET FOR EXP(V22)
40616  01656  41 01777 01660    TESTED ALL NUMBERS?
40617  01657  45 00000 01664    YES: JUMP
40620  01660  11 30000 20000    NO: EXP(V21) ⟶ (A)
40621  01661  42 30000 01655    EXP(V21) < EXP(V22)? YES: TEST EXP V21 < V22
40622  01662  15 01661 01660    NO: L EXP(V22) → L EXP(V21)
40623  01663  45 00000 01655    RETURN TO TEST EXP(V22)
40624  01664  15 01660 01665    STORE
40625  01665  11 30000 00012         D
40626  01666  56 30000 01667    DUMMY
40627  01667  41 00024 01701    READY TO TYPE?
40630  01670  11 01621 00024    YES: RESET TYPING INDEX
40631  01671  61 00000 01757    C.R.
40632  01672  55 00047 10033    ITERATION INDEXX2^33 → (Q)
40633  01673  37 01743 01735    TYPE WITHOUT SPACE
40634  01674  55 00012 10033    D  X2^33 → (Q)
40635  01675  37 01743 01734    TYPE WITH SPACE
```

PX 71900-9-(164)

```
40636  01676  56 30000 01677    DUMMY, (EXIT)
40637  01677  11 00007 20000    LAST ITERATION SWITCH ⟶(A)
40640  01700  46 40442 01701    LAST ITERATION?JUMP TO PART III
40641  01701  23 10000 10000    NO:   0 ⟶(Q)
40642  01702  11 00012 20000       D ⟶(A)
40643  01703  56 10000 01704    STOP TO CHANGE T IN Q
40644  01704  11 10000 20000
40645  01705  47 01744 01706    T CHANGED?
40646  01706  11 00004 20000    NO:  E ⟶(A)
40647  01707  73 01623 00011    STORE E-R/6,   R ⟶(A)
40650  01710  71 20000 01733     -6R ⟶(A)
40651  01711  35 01714 01714    SET SHIFT
40652  01712  55 00011 00017    E-R/6X2^15 ⟶(Q)
40653  01713  21 01714 10000    SET ADDRESS OF T
40654  01714  55 01624 10044    T_E X2^0 ⟶(Q)
40655  01715  51 01772 01637    STORE T
40656  01716  56 30000 01717    DUMMY
40657  01717  11 00012 20000    D ⟶(A)
40660  01720  42 01637 00222    D < T ?  RETURN TO ITERATE
40661  01721  11 01621 20000    NO:TYPING INDEX ⟶(A)
40662  01722  43 00024 01724       (A)=TEMP INDEX?
40663  01723  37 01676 01671    NO:JUMP TO TYPE
40664  01724  13 00007 00007    YES:SET LAST ITERATION SWITCH
40665  01725  75 30000 01727    X
40666  01726  11 30000 30000         ⟶ V1
40667  01727  23 00046 20000    TEST INDEX = 0
40670  01730  23 00024 20000    TYPE INDEX=0
40671  01731  23 00044 20000    ORTHOG INDEX=0
40672  01732  56 30000 00201    RETURN TO ITERATE
40673  01733  77 77777 77771    -6
40674  01734  61 00000 01771    SPACE
40675  01735  11 00041 00010    SET INDEX
```

PX 74900-9-(164)

```
40676  01736  55 10000 00003
40677  01737  51 01773 20000
40700  01740  35 01760 01741
40701  01741  30 00000 00000
40702  01742  41 00010 01736     TYPE
40703  01743  45 00000 30000     EXIT
40704  01744  11 10000 01637     YES T  CHANGED: STORE NEW T
40705  01745  55 10000 00033     TYPE
40706  01746  37 01743 01734       T=ACCURACY
40707  01747  11 00004 20000     E ⟶ (A)
40710  01750  73 01623 00013     STORE E-R/6, R ⟶ A
40711  01751  71 20000 01623      6R
40712  01752  35 01755 01755     SET SHIFT
40713  01753  16 01755 40426     SET
40714  01754  21 40427 00013         L   (T)
40715  01755  55 40433 20000       77X2^6R ⟶ Q
40716  01756  45 00000 40426
40717  01757  00 00000 00045     C.R.
40720  01760  61 00000 01761     TYPE 0
40721  01761  00 00000 00037      0
40722  01762  00 00000 00052      1
40723  01763  00 00000 00074      2
40724  01764  00 00000 00070      3
40725  01765  00 00000 00064      4
40726  01766  00 00000 00062      5
40727  01767  00 00000 00066      6
40730  01770  00 00000 00072      7
40731  01771  00 00000 00004     SPACE
40732  01772  00 00000 00077
40733  01773  00 00000 00007
40734  01774  00 00000 00001     TEST INDEX
40735  01775  00 00002 00002
```

PX 71900-9-(164)

```
40736  01776  00 00000 00000     N-1
40737  01777  00 00000 00000     N-1

40740  00210  00 00000 00000     ORTHOGONALIZATION
40741  00211  11 00317 20000        S.R.
40742  00212  42 00004 00310     E > 37? PUT YY AT END OF ES
40743  00213  11 00305 00200     NO :  RESET ORTHOG INDEX
40744  00214  11 00004 00312     STORE E
40745  00215  23 00312 00042     STORE E-1
40746  00216  75 20000 00220     0
40747  00217  23 30000 20000          → V2
40750 |00220| 75 30000 00222      L EIGENVECTORS
40751  00221  11 30000 30000        MD → ES
40752  00222  11 00306 00313     SET INDEX = L-1
40753  00223  15 00023 00244     STORE L  (Y)
40754  00224  15 00023 00165     SET INNER PRODUCT S.R.
40755 |00225| 56 30001 00226     DUMMY
40756  00226  37 00177 00156     INNER PRODUCT X,Y'
40757  00227  11 00033 00025     FIRST ELEMENT
40760  00230  11 00034 00026              OP 1
40761  00231  75 30002 00233     Y' Y'
40762  00232  11 00322 00027           → OP 2
40763  00233  37 00051 00054     DIV. XY' /Y' Y' = K
40764  00234  11 00031 00310     STORE
40765  00235  11 00032 00311         K1
40766  00236  15 00022 00251     STORE
40767  00237  16 00022 00254         L(V2)
40770  00240  11 00002 00314     SET INDEX = N-1
40771 |00241| 13 00310 00025     -K1
40772  00242  11 00311 00026           → OP 1
40773  00243  75 30002 00245     Y1'
40774  00244  11 30000 00027           → OP 2
```

PX 71900-9-(164)

```
40775  00245  37 00051 00053    MULT•
40776  00246  11 00031 00025     ←K1 Y1'
40777  00247  11 00032 00026           ⟶ OP 1
41000  00250  75 30002 00252    V2
41001  00251  11 30000 00027           ⟶ OP 2
41002  00252  37 00051 00052    ADD V2 ←K1 Y1'
41003  00253  75 30002 00255     STORE
41004  00254  11 00031 30000        IN V2
41005  00255  21 00244 00037    SET
41006  00256  21 00251 00037      FOR
41007  00257  21 00254 00041        NEXT ELEMENT
41010  00260  41 00314 00241    V2 = K1Y'
41011  00261  21 00232 00037    SET FOR L(Y2  Y2)
41012  00262  41 00312 00301    DONE E VECTORS?
41013  00263  75 30002 00265    YES:  ΔX = V2
41014  00264  11 30000 00025     X1 ⟶   OP 1
41015  00265  75 30002 00267     ΔX1
41016  00266  11 30000 00027           ⟶ OP 2
41017  00267  37 00051 00052    ADD
41020  00270  75 30002 00272     X1 +  ΔX1
41021  00271  11 00031 30000           ⟶ V1
41022  00272  21 00264 00037    SET
41023  00273  21 00266 00037     FOR
41024  00274  21 00271 00041          X2
41025  00275  41 00315 00263      X̄ FORMED
41026  00276  56 30001 00277    DUMMY
41027  00277  75 30210 00210    PART II ⟶ ES
41030  00300  11 40210 00210     JUMP TO PART II
41031  00301  41 00313 00225    L VECTORS DONE?
41032  00302  21 00221 00307    YES:SET FOR NEX L VECTORS
41033  00303  56 30001 00304    DUMMY
41034  00304  45 00000 00216    RETURN
```

PX 71900-9-(164)

```
41035  00305  00 00000 00005    ORTHOG INDEX
41036  00306  00 00000 00000    L-1
41037  00307  00 00000 00000    2LNX2^15
41040  00310  23 00306 00042    K  STORAGE       L-2
41041  00311  23 00307 00006                     2LN-2N
41042  00312  15 00316 00232    INDEX = E-1      STORE L ( Y,'  Y')
41043  00313  75 30000 00320    INDEX = L-1    ALL YY
41044  00314  11 41052 02000    INDEX = N-1       → ES
41045  00315  00 00000 00000    N-1
41046  00316  00 02000 00000     L(Y,'Y')ES
41047  00317  00 00000 00037    E > CONSTANT
41050  00320  23 00220 00006    REDUCE REPEAT COMMAND
41051  00321  45 00000 00213
41052  00322  00 00000 00000     INNER PRODUCTS
41053  00323  00 00000 00000      OF EIGENVECTORS
41054  00324  00 00000 00000       STORAGE
41055  00325  00 00000 00000
41056  00326  00 00000 00000
41057  00327  00 00000 00000
41060  00330  00 00000 00000
41061  00331  00 00000 00000
41062  00332  00 00000 00000
41063  00333  00 00000 00000
41064  00334  00 00000 00000
41065  00335  00 00000 00000
41066  00336  00 00000 00000
41067  00337  00 00000 00000
41070  00340  00 00000 00000
41071  00341  00 00000 00000
41072  00342  00 00000 00000
41073  00343  31 46314 63146    M(β)'s
41074  00344  31 46314 63146         = .8
```

PX 71900-9-(164)

```
41075  00345   31 46314 63146

41076          11 40000 00000    STARTER. STORE F1
41077          31 76370 00000    SUM
41100          75 20576 41102         MT AND CARD ROUTINES
41101          32 76371 00000    SUM
41102          75 21300 41104
41103          32 40000 00000      MAIN PROGRAM
41104          11 20000 10000    STORE CHECK SUM
41105          75 30170 01103    STARTER ⟶ ES
41106          11 41110 01010    JUMP TO ES
41107          00 00000 00000

41110  01010   55 10000 00003    TYPE
41111  01011   51 01773 20000
41112  01012   35 01760 01013      CHECK
41113  01013   00 30000 30000
41114  01014   41 01176 01010        SUM
41115  01015   23 10000 20000    CLEAR Q, A
41116  01016   56 00000 01017    STOP TO SET N→(Q), Ē ⟶(A)
41117  01017   75 30421 01021    CONSTANTS, PART II
41120  01020   11 40001 00001         ⟶ ES
41121  01021   11 10000 00005    STORE N
41122  01022   11 20000 00003    STORE Ē
41123  01023   55 10000 00001    STORE
41124  01024   11 10000 00001         2N
41125  01025   55 10000 00017    STORE
41126  01026   11 10000 00006         2NX2^15
41127  01027   11 00005 00002
41130  01030   23 00002 00042    N-1 ⟶(002)
41131  01031   21 00022 00001    STORE
41132  01032   21 00022 00006         L(V2)
```

PX 71900-9-(164)

```
41133  01033  21 00023 00001
41134  01034  21 00023 00006     STORE
41135  01035  21 00023 00001
41136  01036  21 00023 00006         L  (A IN ES)
41137  01037  37 00326 00327     SET UP AX S.R.
41140  01040  75 30003 01042     M(β)'s
41141  01041  11 41073 00410          PART II END
41142  01042  15 00021 00211     STORE L (V1)
41143  01043  16 01164 00211           L(VMD)
41144  01044  15 00304 00233           L(Mβ1)
41145  01045  15 01164 00251           L(VMD)
41146  01046  16 00022 00251           L(V2)
41147  01047  11 00421 10000     MASK ⟶ (Q)
41150  01050  53 00006 00210     SET
41151  01051  53 00006 00250       REPEAT
41152  01052  56 30000 01053         COMMANDS
41153  01053  75 30421 01055     STORE CONSTANTS, PART II
41154  01054  11 00001 40001           MD
41155  01055  75 30113 01057     ORTHOG S.R.
41156  01056  11 40740 00210           ⟶ ES
41157  01057  54 00406 20071     K ⟶ (A)
41160  01060  73 00001 00306     STORE L
41161  01061  71 00306 00006     STORE
41162  01062  11 20000 00307       2NLX 2^15
41163  01063  23 00316 00006     SET L(Y' Y) ES
41164  01064  23 00306 00042     STORE L-1
41165  01065  11 00421 10000     MASK ⟶(Q)
41166  01066  53 00006 00216     SET
41167  01067  53 00307 00220       REPEAT
41170  01070  53 00006 00313         COMMANDS
41171  01071  15 00022 00217     STORE L (V2)
41172  01072  15 01165 00221           L(EVMD)
```

PX 71900-9-(164)

```
41173  01073  16 00023 00221        L(W,S,)
41174  01074  15 00021 00264        L(V1)
41175  01075  15 00022 00266        L(V2)
41176  01076  16 00021 00271        L(V1)
41177  01077  11 00002 00315        N-1
41200  01100  23 00314 00001   STORE L (Y',Y')AT END OF ES
41201  01101  75 30113 01105   STORE ORTHOG
41202  01102  11 00210 40740         → MD
41203  01103  75 30300 01010   PARTS I,III
41204  01104  11 40440 01500     → ES
41205  01105  11 00421 10000   MASK →(Q)
41206  01106  53 00006 01520   SET
41207  01107  53 00006 01522     REPEAT
41210  01110  53 00006 01555
41211  01111  53 00006 01725       COMMANDS
41212  01112  56 30000 01113
41213  01113  16 01166 01515   STORE L(E VALUES)
41214  01114  15 01164 01521        L(VMD)
41215  01115  16 00021 01521        L(V1)
41216  01116  15 00021 01523        L(V1)
41217  01117  16 01165 01523        L(E VECTOR REGION)
41220  01120  15 01165 01556        L(E VECTOR REGION)
41221  01121  16 00021 01556        L(V1)
41222  01122  15 01166 01575        L(E VALUES)
41223  01123  15 01165 01610        L(E VECTORS)
41224  01124  16 00005 01611        N
41225  01125  15 00021 01641        L (M S1)
41226  01126  15 00022 01643        L (MAX)
41227  01127  16 00021 01643        L(M S1)
41230  01130  21 01643 01167        L (EXP  AX) X2^15 + L (EXP S)
41231  01131  15 00022 01660        L(V2)
41232  01132  15 00022 01661        L(V2)
```

PX 71900-9-(164)

```
41233  01133  21 01660 01170        L(V2)+1
41234  01134  21 01661 01170        L(V2)+1
41235  01135  15 01164 01726        L(VMD)
41236  01136  16 00021 01726        L(V1)
41237  01137  11 00002 01776   STORE N-1
41240  01140  11 00002 01777   STORE N-1
41241  01141  56 30000 01142   DUMMY
41242  01142  11 76537 76536   SET MT DUMP TO OMIT BACKING
41243  01143  75 30300 01145   PARTS I, III
41244  01144  11 01500 40440       → MD
41245  01145  16 01175 40000   SET MD START TO INTERMEDIATE START
41246  01146  45 20000 01156   MJ #2 TO SKIP 1's→E V REGION
41247  01147  75 17777 01151   ALL
41250  01150  11 01171 54000
41251  01151  75 17465 01153      1's
41252  01152  11 01171 63777
41253  01153  11 01172 54001
41254  01154  21 01153 00041      → EV REGION
41255  01155  41 01173 01153
41256  01156  11 01174 10000
41257  01157  45 30000 01161   MJ 3 ON TO AVOID DUMP
41260  01160  37 76510 76500   MT DUMP
41261  01161  11 01177 76536   RESET MT DUMP S.R. BACKING
41262  01162  75 30210 76430   JUMP TO DUMP EV'S ON MT
41263  01163  11 40210 00210            PART II → ES
41264  01164  00 41252 41252   L(VMD) X 2^15 + L(VMD)
41265  01165  00 54000 54000   L(E VECTOR REGION ON MD)X2^15 + L(  )
41266  01166  00 53600 53600   L(E VALUES MD)X 2^15 + L(  )
41267  01167  00 00001 00001
41270  01170  00 00001 00000
41271  01171  20 00000 00000     1
41272  01172  00 00000 00001
```

PX 71900-9+164)

```
41273  01173  00 00000 07631    1'S→EV INDEX
41274  01174  00 00000 00000    I
41275  01175  00 00000 40440    L (INTERMEDIATE START)
41276  01176  00 00000 00013    CHECK SUM INDEX
41277  01177  67 01017 00000    MT S.R. BACKING



76365          11 76464 76463  EV MT RESTORE
76366          11 76567 76536
76367          45 00000 76420
76370          23 10000 20000  0 ⟶ (Q)
76371          75 30030 76373  MT S.R.
76372          11 76550 00001    ⟶ ES
76373          37 00017 00001  RESTORE MD FROM MT
76374          71 40001 40003  2NĒ
76375          35 76474 20000  2NĒ + 400₈ ⟶(A)
76376          73 76475 76467  P = 2NĒ +400 /1737₈
76377          21 10000 76465    P + 1 ⟶(Q)
76400          71 10000 76473    (P + 1) × 37₈ × 2¹⁵ ⟶(A)
76401          11 20000 76470  STORE
76402          75 30030 76404  MT S.R.
76403          11 76550 00001       ⟶ ES
76404          16 76471 00014  STORE L(EV'S)
76405          15 76470 00020  SET MT BACKING
76406          11 76467 00030  P ⟶ MT INDEX
76407          45 20000 76415  MJ #2 TO READ FROM MT #2
76410          37 00021 00001  MT #0 READ EV'S, BACK
76411          56 30000 76424  DUMMY
76412          75 31777 76365  RESTORE
76413          11 74001 00001      ES
76414          56 00000 01532  STOP, JUMP TO MS #2 STOP
```

PX 71900-9-(164)

```
76415   11 76466 10000   J ⟶ (Q)
76416   37 00021 00022   MT#2 READ EV'S, BACK
76417   56 30000 76412   RETURN TO RESTORE ES & STOP
76420   75 30210 76414   PART II→ES
76421   11 40210 00210       JUMP TO STOP
76422   75 30200 76434   STORE YY'S
76423   11 41052 53400     FOR MT DUMP
76424   75 30200 76412   RESTORE
76425   11 53400 41052        YY'S
76426   16 76477 41262
76427   45 00000 40000
76430   31 76472 00052
76431   61 00000 20000   TYPE
76432   34 20000 00006      MT
76433   47 76431 76422
76434   71 40001 40003     2NĒ ⟶ (A)
76435   35 76474 20000    2NĒ + 400₈→(A)
76436   73 76475 76467   P=  2NĒ + 400/ 1737₈
76437   21 10000 76465      P + 1 ⟶(Q)
76440   71 10000 76473   (P + 1) X 37₈ X 2^15 ⟶(A)
76441   11 20000 76470   STORE
76442   75 32000 76444   ES
76443   11 00000 74000      ⟶ ES IMAGE
76444   75 30037 76446   MT S.R.
76445   11 76511 00001          ⟶ ES
76446   15 76471 00003   SET L(EV'S)
76447   11 76467 00036   STORE P
76450   15 76470 00026   SET TAPE BACKING
76451   37 00027 00001   MT DUMP ON #0
76452   75 30037 76454   MT S.R.
76453   11 76511 00001      ⟶ ES
76454   15 76471 00003
```

PX 71900-9-(164)

```
76455          11 76467 00036
76456          15 76470 00026
76457          11 76466 10000      2 ⟶ J(Q)
76460          37 00027 00030    MT DUMP ON #2
76461          75 31777 76463    RESTORE
76462          11 74001 00001       ES
76463          37 76463 01541    STOP, JUMP TO MS#2 STOP
76464          56 00000 01532
76465          00 00000 00001
76466          00 20000 00000
76467          00 00000 00000    L  (P)
76470          00 00000 00000    L(BACKING PARAMETER)
76471          00 53400 53400    L (EV REGION)X 2^15 + L(EV REGION)
76472          04 04470 70157    FLEX WORD
76473          00 00037 00000
76474          00 00000 00400
76475          00 00000 01737
76476          67 01017 00000
76477          56 00000 76370

76500          75 32000 76502    MT DUMP & RESTORE S.R.
76501          11 00000 74000    STORE ES IMAGE
76502          75 30037 00001    MT DUMP ROUTINE
76503          11 76511 00001          ⟶ ES, JUMP TO ROUTINE
76504          75 30030 00001    MT RESTORE ROUTINE
76505          11 76550 00001       ⟶ ES, JUMP TO ROUTINE
76506          75 31777 76510    RESTORE
76507          11 74001 00001         ES
76510          56 00000 56510    STOP

76511  00001   45 10000 00030    MJ #1 TO SET MT UNIT
76512  00002   75 31737 00004    MD BLOCK
```

PX 71900-9-(164)

```
76513  00003  11 40000 00041          ⟶ ES
76514  00004  31 00041 00000    CHECK
76515  00005  75 21736 00007      SUM
76516  00006  32 00042 00000        MD BLOCK
76517  00007  11 20000 00040    STORE CHECK SUM
76520  00010  65 00037 00040    MD BLOCK ⟶ MT
76521  00011  67 00037 00000    BACK TAPE 36 BLOCKS
76522  00012  64 00037 00040    MT ⟶ ES MD BLOCK
76523  00013  31 00041 00000    CHECK
76524  00014  75 21736 00016      SUM
76525  00015  32 00042 00000        MD BLOCK
76526  00016  11 20000 20000    STORE CHECK SUM ⟶(A)
76527  00017  43 00040 00023    CHECK SUMS =?
76530  00020  61 00000 00025    NO:TYPE F
76531  00021  67 00037 00000    BACK TAPE
76532  00022  56 30000 00002    MS #3 RETURN TO TRY AGAIN
76533  00023  21 00003 00035    SET FOR NEXT MD BLOCK
76534  00024  41 00036 00002    TRANSFER COMPLETE=(21), TIMES
76535  00025  56 30000 00026    MS#3 JUMP TO RETURN TAPE
76536  00026  67 01017 00000    BACK TAPE TO ORIGIN
76537  00027  56 30000 76506    MS#3 JUMP TO RESTORE ES
76540  00030  53 00034 00021    SET
76541  00031  53 00034 00026      MT
76542  00032  75 10003 00002        COMMANDS
76543  00033  53 00034 00010
76544  00034  00 70000 00000
76545  00035  00 01737 00000
76546  00036  00 00000 00020
76547  00037  00 00000 00000

76550  00001  45 10000 00022    MJ #1 TO SET MT UNIT OTHER THAN 0
76551  00002  64 00037 00040    READ MD BLOCK ⟶ ES
```

PX 71900-9-(164)

```
76552  00003  31 00041 00000     CHECK
76553  00004  75 21736 00006       SUM
76554  00005  32 00042 00000         MD
76555  00006  11 20000 20000           BLOCK
76556  00007  43 00040 00013     CHECK SUMS=?
76557  00010  61 00000 00007     NO:TYPE G
76560  00011  67 00037 00000     BACK TAPE
76561  00012  56 30000 00002     MS #3 RETURN TO TRY AGAIN
76562  00013  75 31737 00015     YES:MD BLOCK
76563  00014  11 00041 40000         → MD
76564  00015  21 00014 00027     SET FOR NEXTMD BLOCK
76565  00016  41 00030 00002     TRANSFER COMPLETE
76566  00017  56 30000 00020     MS #3 JUMP TO RETURN TAPE
76567  00020  67 01017 00000     BACK TAPE TO ORIGINE
76570  00021  56 30000 76506     MS #3 JUMP TO RESTORE ES
76571  00022  53 00026 00002     SET
76572  00023  53 00026 00011       TAPE
76573  00024  53 00026 00020         COMMANDS
76574  00025  45 00000 00002
76575  00026  00 70000 00000
76576  00027  00 00000 01737
76577  00030  00 00000 00020

76600          45 00000 00000     CARD OUTPUT IC 004
76601          16 76614 76610
76602          45 00000 76606
76603          16 76615 76610
76604          45 00000 76606
76605          16 76616 76610
76606          75 31777 76610
76607          11 00001 74001
76610          75 30530 00000
```

PX 71900-9-(164)

76611	11	76620	00526
76612	75	31777	00000
76613	11	74001	00001
76614	00	00000	00744
76615	00	00000	00751
76616	00	00000	00733
76617	56	00000	76605
76620	00	00000	00037
76621	37	77777	77777
76622	00	74000	00000
76623	23	00001	01262
76624	21	00001	01207
76625	21	01262	00536
76626	21	01207	00536
76627	45	00000	01203
76630	00	00000	00144
76631	00	00000	00310
76632	71	00720	30000
76633	15	20000	00566
76634	55	20000	00017
76635	15	10000	00620
76636	75	20044	00546
76637	23	00731	20000
76640	16	00716	00000
76641	31	00723	00001
76642	55	10000	00030
76643	52	00721	20000
76644	54	20000	00002
76645	44	00554	00554
76646	44	00561	00555
76647	17	00000	20000
76650	21	00540	00720

```
76651   16 20000 00560
76652   45 00000 30000
76653   32 00677 00000
76654   17 00000 20000
76655   11 00730 00650
76656   11 00714 00651
76657   37 00560 00566
76660   55 30000 00000
76661   44 00641 00570
76662   55 10000 00013
76663   51 00725 20000
76664   13 20000 20000
76665   35 00715 00621
76666   55 10000 00006
76667   51 00725 00672
76670   55 10000 00006
76671   51 00725 00665
76672   55 10000 00006
76673   51 00725 00713
76674   44 00655 00603
76675   16 00667 00703
76676   37 00676 00605
76677   41 00672 00673
76700   31 00665 00017
76701   37 00706 00704
76702   11 20000 00652
76703   31 00713 00017
76704   37 00706 00704
76705   16 00716 00000
76706   31 00726 00023
76707   73 10000 10000
76710   31 00652 00107
```

PX 71900-9-(164)

76711	35	10000	10000
76712	12	30000	00672
76713	00	30000	30000
76714	35	10000	20000
76715	73	00652	20000
76716	35	20000	00652
76717	37	00676	00626
76720	41	00665	00677
76721	16	00717	00676
76722	41	00713	00660
76723	41	00713	00677
76724	15	00620	00633
76725	55	30000	00000
76726	21	00566	00726
76727	21	00620	00726
76730	16	00560	00676
76731	13	00720	20000
76732	44	00670	00673
76733	37	00560	00570
76734	75	20014	00644
76735	55	00761	00010
76736	75	30003	00646
76737	11	00710	00650
76740	16	00716	00000
76741	43	00711	00556
76742	00	30000	30000
76743	00	30000	30000
76744	00	30000	30000
76745	75	20003	00646
76746	23	00650	00720
76747	37	00703	00604
76750	43	00665	00667

76751	45	00000	00673
76752	31	00721	00000
76753	35	00651	00664
76754	35	00722	00665
76755	55	00650	00000
76756	00	30000	30000
76757	00	30000	30000
76760	33	00720	00001
76761	37	00703	00670
76762	35	00651	00672
76763	55	00650	00000
76764	00	30000	30000
76765	55	00650	00043
76766	44	00675	00676
76767	21	00651	00727
76770	45	00000	30000
76771	31	00652	00002
76772	32	00652	00001
76773	11	20000	00652
76774	34	20000	00044
76775	47	00667	30000
76776	15	20000	00706
76777	54	00720	10000
77000	75	30000	30000
77001	71	20000	00724
77002	77	00000	00774
77003	77	10000	00744
77004	77	10000	00760
77005	00	30000	30000
77006	53	10000	00733
77007	31	00672	00043
77010	00	00000	00540

```
77011   00 00000 00631
77012   00 00000 00001
77013   00 00000 00003
77014   00 00000 00005
77015   00 00000 00010
77016   00 00000 00012
77017   00 00000 00077
77020   00 00001 00000
77021   00 00000 00014
77022   40 00000 00000
77023   71 01206 30000
77024   15 20000 01045
77025   11 10000 01000
77026   11 20000 00775
77027   16 00773 76612
77030   47 00756 00746
77031   31 76600 00017
77032   35 00741 00741
77033   11 74000 01000
77034   21 76600 01213
77035   45 00000 00000
77036   16 00774 76612
77037   37 00743 00737
77040   45 00000 00747
77041   16 01252 01075
77042   45 00000 01001
77043   16 00774 76612
77044   37 00743 00737
77045   31 76600 00017
77046   35 00755 00755
77047   11 74000 20000
77050   11 20000 10000
```

```
77051    21 76600 01213
77052    51 00772 00776
77053    55 10000 00025
77054    51 00772 00777
77055    16 01253 01011
77056    16 01254 01177
77057    51 01245 20000
77060    47 00771 00767
77061    16 01255 01021
77062    16 01252 01075
77063    45 00000 01001
77064    00 00000 77777
77065    00 00000 76617
77066    00 00000 76600
77067    00 00000 00000
77070    00 00000 00000
77071    00 00000 00000
77072    00 00000 00000
77073    15 01000 01007
77074    11 01010 20000
77075    42 01007 01005
77076    21 01000 00530
77077    75 10500 01011
77100    11 01212 01300
77101    00 00000 00000
77102    00 02000 00000
77103    45 00000 01216
77104    71 00776 00777
77105    11 20000 01256
77106    34 01213 00020
77107    11 20000 01257
77110    11 00776 01261
```

PX 71900-9-(164)

```
77111    55 01261 00020
77112    11 00777 01262
77113    45 00000 01023
77114    11 00776 01262
77115    11 01262 01263
77116    21 01262 00536
77117    11 01046 00001
77120    11 01256 20000
77121    35 01246 20000
77122    73 01247 20000
77123    36 01213 01260
77124    11 01256 20000
77125    73 01247 10000
77126    11 20000 01266
77127    36 01213 01267
77130    35 01242 20000
77131    73 01242 01270
77132    23 01270 01213
77133    31 01266 00020
77134    11 20000 01271
77135    31 01266 00001
77136    36 01213 01272
77137    45 00000 01047
77140    00 00000 00140
77141    37 00540 00540
77142    20 00000 00000
77143    37 00540 00540
77144    20 00000 00000
77145    15 01000 01077
77146    15 01000 01107
77147    11 00777 01264
```

PX 71900-9-(164)

DETERMINANT EVALUATION PACKAGE-REAL-CA 010

This is a package including Floating Point Real Arithmetic, CA 001 an interpretive routine for the real arithmetic and the real determinant evaluation routine. It is coded in standard form and can be assembly modified. If this package is assembled at 0100, it will evaluate an N X N determinant for $N = (22)_8 = (18)_{10}$. Each element of the determinant takes two (2) cells, and the entire determinant must be in ES. This routine requires a setting up for particular N, location of determinant in ES, and the location of the result in ES, and can not be used for any other N, determinant location, or result location unless the routine is restored to its original form and set up again. Any number of determinants may be evaluated for a given set-up. The determinant is destroyed during the evaluation. The elimination method is used.

Steps of Elimination:

1. $1 \rightarrow$ (Det. value location).
2. $\frac{a_{ij}}{a_{11}}$ j=2, 3,..., N (if $a_{11} = 0$ exchange row 1 with a row which has a non-zero first element, and change the sign of (Det. value location).
3. $A_{ij} - A_{11} \frac{A_{ij}}{A_{11}}$ j, i = 2, 3,..., N.
4. (Det. value location) $A_{11} \rightarrow$ (Det. value location)
5. Reduce the order of the resulting matrix by one by removing from consideration row one, and col. 1.
6. Repeat steps 2-5 until order of the matrix is reduced to one.

This routine uses 2 N cells (one row) immediately following the determinant as temporary storage. After it is set up, the last $(46)_{10} = (56)_8$ cells of the routine are no longer used (from 1340-1416). The amount of storage needed for the determinant is given by $2(N^2+N)$. The elements of the matrix must be stored by rows.

The real arithmetic, or the real arithmetic and interpretive routine may be used independent of the determinant routine by choosing the proper entrances.

For instructions on the use of real arithmetic see CA 001. The interpretive routine performs the real arithmetic operations referring to a parameter word for the locations of the mantissas of the operands and result.

REAL ARITHMETIC ENTRANCE

Co 37 01001 01002 Add Ent

Co 37 01001 01003 Mult Ent

Co 37 01001 01004 Div Ent

INTERPRETIVE ROUTINE ENTRANCE

Co 37 01001 PPPPP

C_1 AAAA BBBB CCCC

Where operations performed are:

Add (A)+(B)→(C) P=1106

Subt (A)-(B)→(C) P=1110

Mult (A)X(B)→(C) P=1112

Div (A)÷(B)→(C) P=1114

Where AAAA-location of mantissa of 1st operand

BBBB-location of mantissa of 2nd operand

CCCC-location of mantissa of result

DETERMINANT SET-UP ENTRANCE

Co 37 01001 01153

C_1 DDDD RRRR NNNN

Where DDDD-location of Det. in ES

RRRR-location of result in ES

NNNN order of the determinant (octal)

ANALYSIS
PREPARED BY J. M. Ellis
CHECKED BY D. B. Parker
REVISED BY

CONVAIR
A DIVISION OF GENERAL DYNAMICS CORPORATION
SAN DIEGO

DETERMINANT EVALUATION ENTRANCE

Co 37 01001 01153

Permanent const. used 40, 74, 77, 43, 66, 44.

Temp. storage 00003-00032, (A), (Q)

Commands for assembly modification 407_8 $(263)_{10}$

Drum Address-70723-71340

PX 71900-9-(165)

```
70723  01000  37 76000 76002   ALARM EXIT
70724  01001  45 00000 30000   EXIT
70725  01002  45 00000 01027        ADD ENTRANCE
70726  01003  45 00000 01073          MULTIPLY ENTRANCE
70727  01004  11 00027 20000   DIVIDE ENTRANCE M (OP 2)→(A)
70730  01005  47 01006 01000   DIVIDING BY 0?
70731  01006  11 00025 20000   NO: NUMERATOR ⟶(A)
70732  01007  47 01014 01010   NUMERATOR = 0?
70733  01010  11 00040 00031   YES:  0
70734  01011  11 00040 00032     ⟶ ANSWER
70735  01012  37 01012 01013   S.R. EXIT
70736  01013  45 00000 01001   JUMP TO EXIT
70737  01014  54 20000 00042   NO: M (NUMERATOR) X 2^34 ⟶ (A)
70740  01015  73 00027 10000   QUOTIENT X 2^34 ⟶ (Q)
70741  01016  11 00040 00025   0 ⟶ 25
70742  01017  74 10000 00025   NORMALIZE IF (00025) = 0 OR 71
70743  01020  11 20000 00031   NORMALIZED MANTISSA STORED
70744  01021  23 00026 00030   DIFF OF EXP. ⟶ (00026)
70745  01022  11 00025 20000   H ⟶ A
70746  01023  47 01025 01024   H = 0?
70747  01024  21 00026 00074   H = 0 CORRECTED
70750  01025  11 00026 00032   H = 71 EXP ⟶ 00032
70751  01026  45 00000 01012   JUMP TO EXIT
70752  01027  11 00025 20000   ADDITION M (OPERAND 1) ⟶ (A)
70753  01030  47 01031 01051   M (OPERAND 1) = 0?
70754  01031  11 00027 20000   NO: M (OPERAND 2) ⟶ (A)
70755  01032  47 01036 01033   M (OPERAND 2) = 0?
70756  01033  11 00025 00031   YES: M (OPERAND 1) ⟶ M (ANSWER)
70757  01034  11 00026 00032   E (OPERAND 1) ⟶ E (ANSWER)
70760  01035  45 00000 01012   JUMP TO EXIT
70761  01036  11 00026 20000   NO: E (OPERAND 1) ⟶ (A)
```

By: J. H. Ellis
Checked by: D. B. Parker

70762	01037	36	00030	00032	E(OP 1) MINUS E(OP2) = K → (00032)
70763	01040	46	01047	01041	K > 0?
70764	01041	11	00025	20000	YES: EXCHANGE
70765	01042	11	00027	00025	OPERAND
70766	01043	11	20000	00027	ONE
70767	01044	11	00026	20000	AND
70770	01045	11	00030	00260	
70771	01046	11	20000	00030	TWO
70772	01047	12	00032	20000	NO: |K| → (A)
70773	01050	42	01072	01054	|K| < 35?
70774	01051	11	00027	00031	|K| ≥ 35 OR M(OP) 1 = 0
70775	01052	11	00030	00032	(OP 2) → ANSWER
70776	01053	45	00000	01012	JUMP TO EXIT
70777	01054	16	20000	01055	|K| < 35? SET SHIFT OF 5 BITS
71000	01055	54	00027	30000	M(OP2) X $2^{(K)}$ → (A)
71001	01056	35	00025	20000	M(OP2) X 2^{K} + M(OP1) → (A)
71002	01057	11	00040	00032	0 → (00032)
71003	01060	74	20000	00032	NORMALIZE M(ANS) = H → (00032)
71004	01061	11	20000	00031	M(ANS) → (00031)
71005	01062	47	01064	01063	M(ANS) = 0?
71006	01063	13	00032	00026	YES: −H → (00026)
71007	01064	11	00032	20000	NO: H → (A)
71010	01065	42	01071	01067	H < 38?
71011	01066	23	00026	00077	NO: E(OP1) − 32 → (00026)
71012	01067	21	00032	00026	YES: E(OP1) PLUS H (OR H−72)
71013	01070	45	00000	01012	JUMP TO EXIT
71014	01071	00	00000	00*46	DEC 38
71015	01072	00	00000	00*43	DEC 35
71016	01073	71	00025	00027	MULT M(OP1) X M(OP 2) → (A)
71017	01074	11	00040	00025	0 → (00025)

PX 71900-9-(165)

By: J. N. Ellis
Checked by: D. B. Parker

```
71020  01075  74 20000 00025   NORMALIZE M(ANS)=34 OR 35 --> H
71021  01076  11 20000 00031   M(ANS)-->(00031)
71022  01077  47 01102 01100   M(ANS) = 0?
71023  01100  11 00040 00032   YES: 0 = F(ANS)
71024  01101  45 00000 01012   JUMP TO EXIT
71025  01102  23 00025 01072   NO: H-35 --> (00025) = 0 OR -1
71026  01103  21 00026 00030   E(OP1) PLUS E(OP 2)-->(A)-->(00026)
71027  01104  35 00025 00032   E (ANS)->(00032)
71030  01105  45 00000 01012   JUMP TO EXIT
71031  01106  16 01147 01134   ADD
71032  01107  45 00000 01115     ENTRANCE
71033  01110  16 01151 01134   SUBTRACT
71034  01111  45 00000 01115     ENTRANCE
71035  01112  16 01150 01134   MULTIPLY
71036  01113  45 00000 01115       ENTRANCE
71037  01114  16 01146 01134   DIVIDE ENTRANCE
71040  01115  31 01001 00017   SET
71041  01116  15 20000 01117     PARAMETER
71042  01117  11 00000 01152       LOCATION
71043  01120  55 01144 10003   SET
71044  01121  53 01152 01136   L  (Z)
71045  01122  55 01152 00033   SET
71046  01123  55 01145 10003   L  (X)
71047  01124  53 01152 01131
71050  01125  54 01152 00014   SET
71051  01126  53 01152 01133    L(Y)
71052  01127  21 01001 00074    UP EXIT LOC.
71053  01130  75 30002 01132   X ->
71054  01131  11 00000 00025   (OP 1)
71055  01132  75 30002 01134   Y ->
```

PX 71900-9-(165)

By: J. M. Ellis
Checked by: D. B. Parker

```
71056  01133  11 00000 00027    (OP 2)
71057  01134  37 01012 00000    REAL ARITH. JUMP
71060  01135  75 30002 01137    ANSWER
71061  01136  11 00031 00000    → (Z)
71062  01137  37 01137 01140    S.R. EXIT
71063  01140  45 00000 01001    JUMP TO EXIT
71064  01141  13 00027 00027    - (00027)→ (00027)
71065  01142  37 01012 01002    JUMP TO SUB
71066  01143  45 00000 01135    JUMP TO STORE (ANS) + EXIT
71067  01144  70 00000 00777    MASK
71070  01145  00 00777 70000    MASK
71071  01146  00 00000 01004
71072  01147  00 00000 01002
71073  01150  00 00000 01003
71074  01151  00 00000 01141
71075  01152  00 00000
71076  01153  45 00000 01303    SET UP S. R. ENTRY
71077  01154  15 01405 01115    SET PARAMETER LOC
71100  01155  15 01323 01174    SET L (A11) X 2^15 → (01174)
71101  01156  15 01323 01177    SET L (A111) 2^15 → (01177)
71102  01157  11 01322 01225    SET L (A12  A11  A12)
71103  01160  11 01306 01242    SET L (A12  A21  T2)
71104  01161  11 01307 01244    SET L (A22  T2   A22)
71105  01162  11 01321 01255    SET L (T2   A11  A22)
71106  01163  11 00066 00003     1 →
71107  01164  11 00074 00004     (T1)
71110  01165  11 01302 01337    STORE 2X2^24 + 4
71111  01166  11 01314 01336    STORE 2(N-1) · 2^24
71112  01167  75 30006 01171    STORE COPYS
71113  01170  11 01302 01325       2N X. 2^15
```

PX 71900-9-(165)

By: J. N. Ellis
Checked by: D. B. Parker

```
71114  01171  21 01325 01311    N-1
71115  01172  11 01305 01333    INDEX = N-2
71116  01173  11 01327 01334    INDEX = N-1
71117  01174  11 00000 20000    (A11)→A
71120  01175  47 01217 01166     A = 0?
71121  01176  75 30000 01200    YES:  ROW 1
71122  01177  11 00000 00000     → (TEMP ROW)
71123  01200  15 01177 01205    STORE
71124  01201  21 01205 01326       L(ROW 2)
71125  01202  55 01177 20025    STORE
71126  01203  16 10000 01205       L (ROW1)
71127  01204  75 30000 01206    ROW 2
71130  01205  11 00000 00000     → ROW 1
71131  01206  55 01205 20025    STORE
71132  01207  16 10000 01211       L(ROW 2)
71133  01210  75 30000 01212    ROW 1 (TEMP ROW)
71134  01211  11 00000 00000     →  ROW 2
71135  01212  13 00003 00003    (T1)→(T1)
71136  01213  21 01326 01303    2NX2'' + 2NX2°
71137  01214  41 01334 01174    TEST NEW A11
71140  01215  75 10002 01277    COL 1 = 0
71141  01216  11 00040 00003    0 → |DET|    JUMP TO EXIT
71142  01217  11 01303 01326    COL1 = 0   RESTORE CONSTANT
71143  01220  23 01327 00074    INDEX - 1
71144  01221  21 01177 01303    SET FOR
71145  01222  21 01174 01312     NEXT ROW
71146  01223  11 01330 01334    SET INDEX = N-1
71147  01224  37 01137 01114
71150  01225  00 00000          A12/A11 →   (A)
71151  01226  21 01225 01302    SET FOR A13
```

PX 71900-9-(165)

By: J. N. Ellis
Checked by: D. B. Parker

```
71152  01227  41 01334 01224    R/A11
71153  01230  11 01330 01334    SET INDEX=N -2
71154  01231  11 01330 01335    SET INDEX=N-2
71155  01232  31 01242 00003    SET
71156  01233  11 01324 10000    L     ( A21 )
71157  01234  53 20000 01235
71160  01235  11 00000 20000    (A21)→(A )
71161  01236  47 01241 01237    (A21) = 0?
71162  01237  21 01244 01317    YES: SET FOR NEXT ROW
71163  01240  45 00000 01252    JUMP TO SKIP ROW
71164  01241  37 01137 01112    NO:   X
71165  01242  00 00000 00 *     A12 X A21→ T
71166  01243  37 01137 01110
71167  01244  00 00000 00 *-    A22+(A22 A21)→ A22
71170  01245  21 01242 01301    SET FOR A13
71171  01246  21 01244 01302    SET FOR A23
71172  01247  41 01335 01241    0 → 1 COL
71173  01250  23 01242 01336
71174  01251  21 01244 01337
71175  01252  21 01242 01310
71176  01253  41 01334 01231    1ST COL 1, 0,0
71177  01254  37 01137 01112    X
71200  01255  00 00000 00000    (T)   (A11)→ T
71201  01256  23 01330 00074    INDEX -1
71202  01257  23 01336 01301    4(N+1)X2^24 + 4 X 2^12
71203  01260  21 01337 01302    4 · 2^24 4 · 2^12 + 4
71204  01261  21 01255 01311    SET FOR A22
71205  01262  21 01331 01315    SET
71206  01263  11 01331 01242      PARAMETERS
71207  01264  21 01332 01320
```

PX 71900-9-(165)

By: J. N. ELLIS
Checked by: D. B. Parker

71210	01265	11	01332	01244	
71211	01266	21	01325	01309	
71212	01267	21	01225	01325	
71213	01270	41	01333	01173	DET REDUCED ?
71214	01271	11	01255	01273	YES:
71215	01272	37	01137	01112	X
71216	01273	00	00000		T → |DET|
71217	01274	15	01406	01115	RESET INT. S R
71220	01275	15	01406	01127	
71221	01276	75	30002	01001	STORE
71222	01277	11	00003	00000	|DET| & EXIT
71223	01300	00	00000	20000	
71224	01301	00	02000	00000	
71225	01302	00	02000	00002	
71226	01303	31	01001	00017	S R SET UP ENT
71227	01304	15	20000	01305	SET (L) PARAMETER
71230	01305	11	00000	01416	PARAMETER → (T)
71231	01306	21	01001	00074	EXIT PLUS 1
71232	01307	11	01410	10000	MASK → (0)
71233	01310	51	01416	01153	STORE N
71234	01311	11	01153	01304	STORE
71235	01312	23	01304	00074	N-1
71236	01313	36	00074	01305	STORE N-2
71237	01314	71	01300	01153	STORE
71240	01315	11	20000	01310	$2N \cdot 2^{12}$
71241	01316	55	01310	10003	STORE
71242	01317	11	10000	01303	$2N \cdot 2^{15}$
71243	01320	35	01300	01311	STORE $2(N+1) \cdot 2^{12}$
71244	01321	54	20000	00003	STORE
71245	01322	11	20000	01312	$2(N+1) \cdot 2^{15}$

PX 71900-9-(165)

By: J. N. Ellis
Checked by: D. B. Parker

71246	01323	11	01153	01313	STORE
71247	01324	21	01313	00074	N + 1
71250	01325	71	01301	01153	STORE
71251	01326	11	20000	01314	2NX 2^{24}
71252	01327	15	01409	01127	SET S R
71253	01330	35	01301	01315	STORE 2 (N+1) · 2^{24}
71254	01331	11	01311	01316	STORE
71255	01332	21	01316	01302	2X2^{24} 2(N + 1) · 2^{12} + 2
71256	01333	21	01315	01311	STORE 2(N+1) 2^{24} 2(N + 1) · 2^{12}
71257	01334	71	00041	01153	STORE
71260	01335	35	01314	01317	2NX 2^{24} 2N
71261	01336	35	01302	01320	STORE 2(N+1) 2^{24} 2(N + 1)
71262	01337	23	01314	01301	STORE 2(N-1) · 2^{24}
71263	01340	11	01412	10000	MASK → (Q)
71264	01341	11	01414	01306	STORE L (T2)
71265	01342	11	01414	01307	STORE L (T2)
71266	01343	53	01416	01306	STORE
71267	01344	53	01416	01307	A11
71270	01345	55	01416	00030	
71271	01346	11	01411	10000	MASK → Q
71272	01347	53	01416	01306	STORE L (A11) X·2^{12}
71273	01350	11	01415	01321	STORE L (T1)·2^{24} + L (T1)
71274	01351	53	01416	01321	STORE L(A11)·2^{12}
71275	01352	11	01410	10000	MASK → (Q)
71276	01353	53	01416	01277	STORE L(|DET|)
71277	01354	55	01416	00030	
71300	01355	11	01410	10000	MASK → (Q)
71301	01356	11	01306	01322	
71302	01357	53	01416	01322	STORE
71303	01360	53	01416	01307	L (A11)

PX 71900-9-(165)

By: J. N. Ellis
Checked by: D. B. Parker

```
71304  01361  55 01416 00017
71305  01362  11 01407 10000    MASK ⟶ (Q)
71306  01363  51 01416 01323    STORE L (A11)  2^15
71307  01364  21 01322 01302    STORE L (A12 A11 A12)
71310  01365  21 01306 01310
71311  01366  21 01306 01301    STORE L (A12 A21 T2)
71312  01367  21 01307 01320    STORE L (A22 T2 A22)
71313  01370  11 01413 10000    MASK⟶ (Q)
71314  01371  53 01303 01176    SET
71315  01372  53 01303 01204     REPEAT
71316  01373  53 01303 01210       COMMANDS
71317  01374  15 01323 01211    STORE L (A11)
71320  01375  71 01303 01153    2N^2 ⟶ A
71321  01376  35 01211 01211    STORE L (T ROW)  2^15
71322  01377  55 01211 20025
71323  01400  16 10000 01177
71324  01401  11 01404 01153    SET ENT JUMP
71325  01402  11 01407 01324    STORE MASK
71326  01403  45 00000 01001    JUMP TO EXIT
71327  01404  45 00000 01154
71330  01405  00 01137 00000
71331  01406  00 01001 00000
71332  01407  00 07777 00000
71333  01410  00 00000 07777
71334  01411  00 00777 70000
71335  01412  77 77000 00000
71336  01413  00 00777 00000
71337  01414  00 00000 70007
71340  01415  00 03000 00003
```

PX 71900-9-(165)

FOUR POINT LAGRANGE INTERPOLATION FOR BIVARIATE FUNCTIONS OR THEIR DERIVATIVES (FIXED POINT)

I. DESCRIPTION:

If from one to five quantities F^n are each functions of two variables x and y and are given in the form of tabled values (not necessarily at equal distance points); then this subroutine evaluates these functions or one of the first partial derivatives of these functions by interpolation. The nearest tabled values of x and y are first found. The subroutine uses tallies γ and ρ to signify displacement in the x and y tables. $\gamma = 0$ signifies the first four x values (x_0, x_1, x_2, x_3). $\rho = 0$ signifies the first four y values (y_0, y_1, y_2, y_3). A sixteen point grid of function values corresponds to the selected values of x's and y's. The Lagrange method for interpolation is used to evaluate the functions at the given point (x, y) for each F^n.

The subroutine is also set up to compute the partial derivatives of the functions by the use of control words which are specified as follows:

04	44400	00000	compute $\frac{\partial F}{\partial x} \cdot \Delta$
00	00044	44000	compute $\frac{\partial F}{\partial y} \cdot \Delta$
00	00000	00000	compute F

In order to eliminate scaling problems, the subroutine computes the partial derivatives times an "increment" Δ' (stored at the same scale factor as the function). The coder must divide this by Δ' (which is stored in cell 00001 with the same scaling as the independent variable involved).

The subroutine uses a normalization process to achieve full accuracy without overflow. Unless the point x, y lies outside the region, no overflow can occur.

If the point x, y lies within the same rectangle of the xy grid as the previously used point, much of the computation is not repeated with a saving of over half in time.

II. SPECIFICATIONS:

A ----------Locations and Storage

1. The subroutine is stored in cells 01000 - 01246 using $(2\ 4\ 7)_8$ or $(1\ 6\ 7)_{10}$ cells

2. E S Temporaries used are cells 00001 to 00072 plus $(2\ 0\ )_8$ cells for each dependent function.

3. Cells to be modified: $(247)_8 = (167)_{10}$

4. Constants not modified: none

5. Information stored on drum for each set of tables:

(a) Entrance and exit code: 5 cells

(b) Parameters: 7 cells

(c) Temporaries: $(24)_{10}$ cells

(d) Tables of the independent variables x and y and the dependent variables F^n.

B ----------Code, Parameters and Temporaries for each set of Tables:

1. A starter program followed by current grid information for each set of from 1 to 5 functions is stored in MD. This optional MD location is now referred to as cells d thru (d + 50)

2. The subroutine is entered by the command:

37 d (d + 2)

3. The information is laid out as follows:

CELL ADDRESS	CONTENT	EXPLANATION
d	75 3 (p) (30000)	Save current data
d + 1	11 00030 d + 5	p = $(44)_8$ + K $(20)_8$

PX 71900-9-(166)

ANALYSIS
PREPARED BY C. J. SWIFT, M. COVHER
CHECKED BY D. SHUMWAY
REVISED BY

CONVAIR
A DIVISION OF GENERAL DYNAMICS CORPORATION
SAN DIEGO

CELL ADDRESS		CONTENT		EXPLANATION
d + 2	75	3 (p+1)	(01002)	Load current data
d + 3	11	d + 4	00027	
d + 4	45	00000	d	Return from subroutine
d + 5	00	M	00000	M = number of x values
d + 6	00	N	00004	N = number bf y values
d + 7	(77	77776	77777)	γ (initial γ = -1)
d + 10	(00	00000	00000)	ρ
d + 11	00	(k + 1 at 2^{19})	00000	K = number of dependent variables tabled
d + 12	00	A_x	00000	A_x = address of x table
d + 13	00	A_y	00000	A_y = address of y table
d + 14	00	A_F^1	00000	A_F^n = address of F^n table
d + 15	00	A_F^2	00000	
.....				
d + 20	00	A_F^5	00000	
d + 21				Temporary storage (current x, y and intermediate computations that are saved)
.....				
d + 50				
d + 50 + $(20)_8$ K				

C Table Format

The x and y tables must be in monotonic ascending order. The functions F^n are stored column wise each column corresponding to one value of X.

D – Results: — The results are stored as follows:

(00001) Δ' when partial differentiation occurs

(00002) $F_1(x,y)$ or $\Delta \cdot \left(\frac{\partial F}{\partial x}\right)$ or $\Delta \cdot \left(\frac{\partial F_1}{\partial y}\right)$

(00003) $F_2(x,y)$ or $\Delta \cdot \left(\frac{\partial F^2}{\partial x}\right)$ or $\Delta \cdot \left(\frac{\partial F_2}{\partial y}\right)$

(00004) $F_3(x,y)$ or $\Delta \cdot \frac{\partial F_3}{\partial x}$ or $\Delta \cdot \frac{\partial F_3}{\partial y}$

(00005) $F_4(x,y)$ or $\Delta \cdot \frac{\partial F_4}{\partial x}$ or $\Delta \cdot \frac{\partial F_4}{\partial y}$

(00006) $F_5(x,y)$ or $\Delta \cdot \frac{\partial F_5}{\partial x}$ or $\Delta \cdot \frac{\partial F_5}{\partial y}$

BIVARIATE INTERPOLATION

```
01000  37 76000 76002    ALARM EXIT
01001  45 00000 30000    EXIT
01002  16 00027 01001    SET UP EXIT
01003  11 00032 20000    COMPARE SIGMA
01004  46 01030 01005       WITH ZERO
01005  11 00024 20000    TEST FOR X
01006  42 00045 01023      AT CENTER
01007  42 00046 01014        OF GRID
01010  11 00032 20000    TEST FOR
01011  36 00030 20000     UPPER
01012  35 01230 20000      LIMIT OF
01013  46 01030 01014       X TABLE
01014  11 00025 20000    TEST FOR Y
01015  42 00051 01025      AT CENTER
01016  42 00052 01055        OF GRID
01017  11 00033 20000    TEST FOR
01020  36 00031 20000     UPPER
01021  35 01230 20000      LIMIT OF
01022  46 01033 01055       Y TABLE
01023  11 00032 20000    TEST FOR
01024  47 01027 01014     ZERO SIGMA
01025  11 00033 20000    TEST FOR
01026  47 01032 01055     ZERO RHO
01027  23 00032 01233    SIGMA MINUS TWO
01030  21 00032 01234    SIGMA PLUS ONE
01031  45 00000 01034    JUMP TO NEW VALUES
01032  23 00033 01233    RHO MINUS TWO
01033  21 00033 01234    RHO PLUS ONE
01034  37 01055 01110    SET SWITCH AND JUMP
01035  16 01226 01175    SET UP FOR
01036  15 01233 01174     DENOMINATOR
01037  37 01167 01155    JUMP TO PQ SUBR
```

BIVARIATE INTERPOLATION

```
01040  16 01227 01175     PARTIAL SWITCH TO NUM
01041  16 01237 01044     ZERO SET TO J AND I
01042  16 01240 01046     RIJ SET TO ROO
01043  15 01237 01044     PI SET TO PO
01044  71 30015 30011     QJ
01045  54 20000 00047       TIMES
01046  11 20000 30054         PI
01047  21 01046 01235     IJ PLUS ONE
01050  43 01241 01055     JUMP TO NUM AFTER LAST IJ
01051  21 01044 01234     J PLUS ONE
01052  42 01242 01044     TEST FOR LAST J
01053  21 01044 01235     I PLUS ONE
01054  45 00007 01043     INTO LOOP AGAIN
01055  37 01055 30000     NUM DEN SWITCH
01056  15 01054 01174     SET UP FOR NUM
01057  45 00000 01060     JUMP TO PQ SUBR
01060  37 01167 01155     SET PI TO PO
01061  16 01237 01066     SET I
01062  75 10005 01064     PICK UP
01063  11 01236 00002       ZERO TO SUMS
01064  15 01240 01067     OO TO IJ
01065  15 01237 01066     SET J
01066  71 30015 30011     QJ TIMES PI
01067  73 30054 00007      DIVIDED BY RIJ
01070  11 01246 01076
01071  11 01232 00010     SET UP N INDEX
01072  15 01067 01074     SET
01073  21 01074 01232       FN-IJ
01074  71 30074 00007     TIMES FN-IJ
01075  54 20000 00047       PLUS
01076  35 30002 30002         FN SUM
01077  21 01076 01231     IJK PLUS ONE
```

PX 71900-9-(166)

BIVARIATE INTERPOLATION

```
01100  21 00010 01232     N PLUS ONE
01101  42 00034 01073     TEST FOR LAST N
01102  21 01067 01234     RIJ PLUS ONE
01103  21 01066 01234     QIJ PLUS ONE
01104  42 01242 01066     5 COMPARED TO J
01105  21 01066 01235     PI PLUS ONE
01106  42 01243 01065     5 COMPARED TO I
01107  45 00037 01001     OUT OF INTERPOLATION
01110  16 01244 01131     SET COORD SWITCH
01111  15 00035 01133     SET FIRST
01112  21 01133 00032       X ADDRESS
01113  15 00036 01135     SET FIRST
01114  21 01135 00033       Y ADDRESS
01115  54 00031 10071     COMPUTE
01116  16 01240 01140      ADDRESS
01117  15 01107 01121       OF
01120  11 01232 00003        FIRST
01121  15 30000 01140         ROW
01122  21 01140 01154          AND
01123  72 10000 00032           FIRST
01124  35 00033 01140             FUNCTION
01125  45 00000 01126     SET TRANSFERS
01126  35 00031 01142       FOR
01127  35 00031 01144         F
01130  35 00031 01146         PICK UPS
01131  45 00000 30000     COORD SWITCH
01132  75 30004 01134     PICK UP
01133  11 30000 00044       X S
01134  75 30004 01136     PICK UP
01135  11 30000 00050       Y S
01136  37 01131 01137     SWITCH
01137  75 30004 01141     PICK UP FN S
```

PX 71900-9-(166)

BIVARIATE INTERPOLATION

```
01140  11 30000 00054     FN
01141  75 30004 01143      FUNCTION
01142  11 30000 00100
01143  75 30004 01145        G
01144  11 30000 00104         R
01145  75 30004 01147          I
01146  11 30000 00110           D
01147  45 00000 01150
01150  21 01121 01234     ADDRESS F
01151  21 00003 01232       PLUS ONE
01152  42 00034 01121       REPEAT LOOP
01153  45 00000 01003     JUMP TO BEGINNING
01154  00 00000 00020     CONSTANT
01155  11 00024 00007     W SET TO X
01156  75 30004 01160     WN SET
01157  11 00044 00002       TO XI
01160  37 01225 01170     DO INNER LOOP
01161  75 30004 01163     P S
01162  11 00015 00011       IN PLACE
01163  11 00025 00007     W SET TO Y
01164  75 30004 01166     WN SET
01165  11 00050 00002       TO YJ
01166  37 01225 01170     JUMP TO INNER LOOP
01167  45 00000 30000     EXIT SUBR
01170  11 00005 20000     W3 MINUS WO
01171  36 00002 00010      EQUALS DELTA
01172  16 01243 01217     PRESET
01173  75 10003 01175     COMPUTE
01174  11 30000 00021      W MINUS WN
01175  75 30003 01212       OR WO MINUS WN
01176  23 00021 00003         GIVING    FACTORS
01177  55 00026 00003     CONTROL WORD TO Q
```

BIVARIATE INTERPOLATION

```
01200  44 01201 01212     TEST FOR PARTIAL
01201  11 00010 00001     STORE DELTA FOR PARTIAL
01202  54 00021 00041     COMPUTE
01203  73 00010 00021      SUM OF
01204  54 00022 20041       PRODUCTS
01205  73 00010 10000        FOR
01206  71 10000 00023          PARTIAL FACTOR
01207  72 00021 00022           AND
01210  72 10000 00023            NORMALIZE
01211  45 00000 01217     JUMP TO STORE
01212  54 00021 00041     COMPUTE
01213  73 00010 20000      TRIPLE
01214  71 20000 00022       PRODUCT
01215  73 00010 20000        AND
01216  71 20000 00023         NORMALIZE
01217  73 00010 30015     STORE QJ
01220  11 00002 00006     ROTATE
01221  75 30004 01223      THE
01222  11 00003 00002       WN
01223  21 01217 01235     J PLUS ONE
01224  42 01245 01173     TEST FOR LAST J
01225  45 00000 30000     EXIT INNER LOOP
01226  00 00000 01212     C
01227  00 00000 01177      O
01230  00 00004 40000       N
01231  00 00001 00001        S
01232  00 00020 00000         T
01233  00 00002 00000          A
01234  00 00001 00000           N
01235  00 00000 00001            T
01236  00 00000 00*-               S
01237  00 00015 00011
```

PX 71900-9-(166)

BIVARIATE INTERPOLATION

```
01240  00 00054 00054
01241  11 20000 00074
01242  71 00021 00000
01243  71 00021 00015        CONSTANTS
01244  00 00000 01132
01245  73 00010 00021
01246  35 00002 00002
```

ANALYSIS
PREPARED BY D. SHUMWAY
CHECKED BY C. J. SWIFT
REVISED BY

CONVAIR
A DIVISION OF GENERAL DYNAMICS CORPORATION
SAN DIEGO

FOUR POINT LAGRANGE INTERPOLATION FOR TRIVARIATE FUNCTIONS OR THEIR DERIVATIVES (FIXED POINT)

I. SPECIFICATIONS:

STORAGE: The sub-routine is stored in cells 01000 - 01305 and may be relocated. ES working storage 00002 - 00363 is used. Tables of any size are normally on MD.

TIME: Approx. 0.3 sec. minimum + 0.15 sec. for each change in 64 point grid.

METHOD: Third order interpolation is effected by the use of the Lagrange 4 point interpolation formula for 3 variables. This passes a function through a 64 pt. cubic array or grid of points chosen from the xyz space of the table.

II. DESCRIPTION:

If two quantities, F_1 and F_2, are functions of three variables, x, y, z and their values are given in tabular form (not necessarily at equal intervals), this subroutine can evaluate F_1 and F_2 or one of their partial derivatives as specified by a control word. The 64 pt. grid is first selected so that the interpolated point (x, y, z) is as close to the center of the grid as possible. This grid is then used for interpolating F_1 and F_2 (or specified partial derivative of same) in parallel.

All factors in the Lagrangian coefficients are normalized to values ≤ 1 to obviate overflow difficulties. If extrapolation is attempted too far outside of the outermost grid, a divide fault will occur.

III. LOCATION AND STORAGE:

The main sub-routine is coded to operate in cells 01000 - 01265. It uses cells 01266 - 01305 for constants and cells 00002 - 00363 in ES for temporaries.

A starter program followed by current grid information for each set of two functions is stored in $(333)_8$ cells of MD. This optional MD location is now referred to as cells d thru d + 332. Initial set up of d thru d + 16 is as follows:

d	75	30326	(30000)	
d + 1	11	00036	d + 5	
d + 2	75	30327	01002	
d + 3	11	d + 4	00035	
d + 4	45		d	
d + 5	00	00000	m	m = no. of values in x table
d + 6	00	00000	n	n = no. of values in y table
d + 7	00	2ℓ	00004	ℓ = no. of values in Z table
d + 10	77	77776	77777	σ = index of current location in x table, initially = 1
d + 11	0	———————	0	ρ = index of current location in y table, initially = 0
d + 12	0	———————	0	τ = index of current location in z table initially = 0
d + 13	00	A_x	00000	A_x= location of 1st x table value
d + 14	00	A_y	00000	A_y= location of 1st y table value
d + 15	00	A_z	00000	A_z= location of 1st z table value
d + 16	11	A_f	00164	A_f= location of 1st F_1 table value

Information in d + 17 - d + 332 is generated by the sub-routine. In order to minimize access time, the tables should be stored in the following sequence:

1. All x values in ascending order
2. All y values in ascending order
3. All z values in ascending order
4. All F_1 and F_2 values alternate, $F_1(x_i, y_j, z_k)$, $F_2(x_i, y_j, z_k)$, - $F_1(x_i, y_j, z\ k+1)$, $F_2(xi, yj, xk+1)$, etc. with i the major, j the

intermediate and k the minor ordering subscript.

Coordinates of the point to be interpolated AND a control word are entered:

(00031) = X

(00032) = Y

(00033) = Z

(00034)= Control word.

For computing F_1 and F_2, the control word = 00 0000000000

For computing $\Delta \cdot \frac{\partial F_1}{\partial x}$ and $\Delta \cdot \frac{\partial F_2}{\partial x}$ the control word = 04444000000

For computing $\Delta \cdot \frac{\partial F_1}{\partial y}$ and $\Delta \cdot \frac{\partial F_2}{\partial y}$ the control word= 0000044440000

For computing $\Delta \cdot \frac{\partial F_1}{\partial z}$ and $\Delta \cdot \frac{\partial F_2}{\partial z}$ the control word = 4000000004444

Entry to the subroutine is programmed: 37 d+2 d.

RESULTS:

(00002) Δ (in case of partial differentiation)

(00003) F_1 or $\Delta \cdot \frac{\partial F_1}{\partial x}$ or $\Delta \cdot \frac{\partial F_1}{\partial y}$ or $\Delta \cdot \frac{\partial F_1}{\partial z}$

(00004) F_2 or $\Delta \cdot \frac{\partial F_2}{\partial x}$ or $\Delta \cdot \frac{\partial F_2}{\partial y}$ or $\Delta \cdot \frac{\partial F_2}{\partial z}$

WHERE (00002) has the scaling of the independent variable

(00003) has the scaling of the F_1

(00004) has the scaling of the F_2

The main subroutine is coded in standard form from 01000 and can be assembly modified.

Subroutine length (including constants)	----	$(306)_8 = (198)_{10}$
ES Temporaries required	----	$(362)_8 = (242)_{10}$
Total ES working space	----	$(670)_8 = (440)_{10}$
No. of words for assembly mod.	----	$(301)_8 = (193)_{10}$

3 VARIABLE INTERPOLATION

```
01000  37 76000 76002     ALARM EXIT
01001  45 00000 30000     EXIT
01002  16 00035 01001     SET UP EXIT
01003  11 00041 20000     COMPARE SIGMA
01004  46 01041 01005      WITH ZERO
01005  11 00031 20000     TEST FOR X
01006  42 00051 01032      AT CENTER
01007  42 00052 01014       OF GRID
01010  33 00036 00017     TEST FOR
01011  35 00041 20000      UPPER
01012  35 01266 20000       LIMIT OF
01013  46 01041 01014         X TABLE
01014  11 00032 20000     TEST FOR Y
01015  42 00055 01034      AT CENTER
01016  42 00056 01023       OF GRID
01017  33 00037 00017     TEST FOR
01020  35 00042 20000      UPPER
01021  35 01266 20000       LIMIT OF
01022  46 01044 01023         Y TABLE
01023  11 00033 20000     TEST FOR Z
01024  42 00061 01036      AT CENTER
01025  42 00062 01077       OF GRID
01026  11 00043 20000     TEST FOR
01027  32 01266 00001      UPPER
01030  36 00040 20000       LIMIT OF
01031  46 01047 01077         Z TABLE
01032  11 00041 20000     TEST FOR
01033  47 01040 01014         ZERO SIGMA
01034  11 00042 20000     TEST FOR
01035  47 01043 01023         ZERO RHO
01036  11 00043 20000     TEST FOR
01037  47 01046 01077         ZERO TAU
```

3 VARIABLE INTERPOLATION

```
01040  23 00041 01303    SIGMA MINUS TWO
01041  21 00041 01304    SIGMA PLUS ONE
01042  45 00000 01050    JUMP TO NEW VALUES
01043  23 00042 01303    RHO MINUS TWO
01044  21 00042 01304    RHO PLUS ONE
01045  45 00000 01050    JUMP TO NEW VALUES
01046  23 00043 01303    TAU MINUS TWO
01047  21 00043 01304    TAU PLUS ONE
01050  37 01077 01137    SET SWITCH AND JUMP
01051  16 01267 01235    SET UP FOR
01052  15 01231 01234       DENOMINATOR
01053  37 01227 01210    JUMP TO PQS SUBR
01054  16 01275 01235    PARTIAL SWITCH TO NUM
01055  16 01270 01060    PI SET TO PO
01056  16 01271 01066    RIJK SET TO ROOO
01057  15 01270 01060    QJ SET TO QO
01060  71 30000 30000    QJ
01061  54 20000 00047     TIMES
01062  11 20000 10000       PI
01063  16 01272 01064    SK SET TO SO
01064  71 10000 30000    QJPI TIMES
01065  54 20000 00047     SK EQUALS
01066  11 20000 30000       RIJK
01067  21 01066 01305    IJK   PLUS ONE
01070  43 01273 01077    JUMP TO NUM AFTER LAST IJK
01071  21 01064 01305    K  PLUS  ONE
01072  42 01274 01064    TEST FOR LAST K
01073  21 01060 01304    J PLUS ONE
01074  42 01275 01060    TEST FOR LAST J
01075  21 01060 01305    I PLUS ONE
01076  45 00000 01057    INTO LOOP AGAIN
01077  37 01077 30000    NUM-DEN SWITCH
```

PX 71900-9-(167)

3 VARIABLE INTERPOLATION

```
01100  15 01272 01234     SET UP FOR NUM
01101  37 01227 01207     JUMP TO PQS SUBR
01102  16 01270 01110     SET PI TO PO
01103  23 00003 20000     ZERO TO
01104  11 20000 00004       SUM CELLS
01105  16 01273 01116     SET FI-IJK TO FI-000
01106  15 01271 01115     SET RIJK TO ROOO
01107  15 01270 01110     SET QJ TO QO
01110  71 30000 30000     QJ
01111  54 20000 00046       TIMES
01112  11 20000 00005         PI
01113  16 01272 01114     SET SK TO SO
01114  71 00005 30000     QJPI TIMES SK
01115  73 30000 10000     DIVIDE BY RIJK
01116  71 10000 30000     TIMES  F1-IJK
01117  54 20000 00050       PLUS
01120  35 00003 00003         F1 SUM
01121  21 01116 01305     1-IJK  PLUS  ONE
01122  11 20000 01123       TO GIVE 2-IJK
01123  30 30000 30000     TIMES F2-IJK
01124  54 20000 00050       PLUS
01125  35 00004 00004         F2 SUM
01126  21 01116 01305     NEXT F1
01127  43 01276 01001     EXIT AFTER LAST F
01130  21 01115 01304     IJK PLUS ONE
01131  21 01114 01305     K PLUS ONE
01132  42 01277 01114     TEST FOR LAST K
01133  21 01110 01304     J PLUS ONE
01134  42 01275 01110     TEST FOR LAST J
01135  21 01110 01305     I PLUS ONE
01136  45 00000 01107     INTO LOOP AGAIN
01137  16 01300 01162     SET COORD SWITCH
```

3 VARIABLE INTERPOLATION

```
01140  15 00044 01164     SET FIRST
01141  21 01164 00041       X ADDRESS
01142  15 00045 01166     SET FIRST
01143  21 01166 00042       Y ADDRESS
01144  15 00046 01170     SET FIRST
01145  21 01170 00043       Z ADDRESS
01146  11 01266 00006     SET LAYER INDEX
01147  11 00047 01173     COMPUTE
01150  54 00040 20070      ADDRESS OF
01151  71 20000 00042       FIRST FUNCTION
01152  35 00043 00010        ON
01153  71 10000 00037          FIRST
01154  71 20000 00041           YZ
01155  32 00010 00001             LAYER
01156  35 01173 01173     FOR FIRST
01157  35 00040 01175       SECOND
01160  35 00040 01177         THIRD
01161  35 00040 01201           FOURTH ROW
01162  45 00000 30000     COORD SWITCH
01163  75 30004 01165     PICK UP
01164  11 30000 00050       X COORD
01165  75 30004 01167     PICK UP
01166  11 30000 00054       Y COORD
01167  75 30004 01171     PICK UP
01170  11 30000 00060       Z COORD
01171  37 01162 01172     SET COORD SWITCH
01172  75 30010 01174     PICK UP
01173  30 30000 30000       FIRST ROW
01174  75 30010 01176     PICK UP
01175  30 30000 30000       SECOND ROW
01176  75 30010 01200     PICK UP
01177  30 30000 30000       THIRD ROW
```

PX 71900-9-(167)

3 VARIABLE INTERPOLATION

```
01200  75 30010 01202     PICK UP
01201  30 30000 30000       FOURTH ROW
01202  23 00006 01304     TEST FOR
01203  42 01304 01003       LAST LAYER
01204  31 10000 00013     SPACING TO
01205  32 01305 00005       NEXT LAYER
01206  45 00000 01156     INTO LOOP AGAIN
01207  11 00031 00010     W SET TO X
01210  75 30004 01212     WN SET TO
01211  11 00050 00003       XI
01212  37 01265 01230     JUMP TO INNER LOOP
01213  75 30004 01215     PI IN
01214  11 00022 00012       PLACE
01215  11 00032 00010     W SET TO Y
01216  75 30004 01220     WN SET
01217  11 00054 00003       TO YJ
01220  37 01265 01230     JUMP TO INNER LOOP
01221  75 30004 01223     QJ IN
01222  11 00022 00016       PLACE
01223  11 00033 00010     W SET TO Z
01224  75 30004 01226     WN SET
01225  11 00060 00003       TO ZK
01226  37 01265 01230     JUMP TO INNER LOOP
01227  45 00000 30000     EXIT PQS SUB
01230  11 00006 20000     W3 MINUS WO
01231  36 00003 00011       EQUALS DELTA
01232  16 01272 01257     SET SK TO S0
01233  75 10003 01235     COMPUTE
01234  11 30000 00026      W MINUS WN
01235  75 30003 30000       OR WO MINUS WN
01236  23 00026 00004         GIVING FACTORS
01237  55 00034 00003     CONTROL WORD TO Q
```

3 VARIABLE INTERPOLATION

```
01240  44 01241 01252     TEST WHETHER PARTIAL
01241  11 00011 00002     STORE DELTA FOR PARTIAL
01242  54 00026 00041     COMPUTE
01243  73 00011 00026      SUM OF
01244  54 00027 20041       PRODUCTS
01245  73 00011 10000        FOR
01246  71 10000 00030          PARTIAL FACTOR
01247  72 00026 00027           AND
01250  72 10000 00030             NORMALIZE
01251  45 00000 01257     JUMP TO STORE
01252  54 00026 00041     COMPUTE
01253  73 00011 10000      TRIPLE
01254  71 10000 00027       PRODUCT
01255  73 00011 10000        AND
01256  71 10000 00030          NORMALIZE
01257  73 00011 30000     STORE SK
01260  11 00003 00007     ROTATE
01261  75 30004 01263      THE
01262  11 00004 00003       WN
01263  21 01257 01305     K PLUS ONE
01264  42 01302 01233     TEST FOR LAST K
01265  45 00000 30000     EXIT INNER LOOP
01266  00 00004 40000     C
01267  00 00000 01252      O
01270  00 00016 00012       N
01271  00 00064 00064        S
01272  00 00010 00022         T
01273  11 20000 00164          A
01274  71 10000 00026           N
01275  71 00021 01237            T
01276  71 10000 00364             S
01277  71 00005 00026
```

PX 71900-9-(167)

3 VARIABLE INTERPOLATION

```
01300  00 00000 01163
01301  11 00164 00164
01302  73 00011 00026
01303  00 00002 00000
01304  00 00001 00000
01305  00 00000 00 *1
```

ORGANIZATION OF DEPARTMENT 66-10

I. GENERAL OBJECTIVES

A. The ultimate aim is to establish Department 66-10 as one of the leading computing and data reduction centers in this country. As a consequence, the Missile Systems Division will have at its disposal an important and profitable asset in meeting its obligations, in increasing its reputation and in attracting business. Furthermore, the Lockheed Aircraft Corporation will be in an even better position to meet its commitments, to increase its productivity, to enhance its reputation and to better its competitive position.

B.

1. Department 66-10 shall perform all MSD work falling within the scope of its activities. It will undertake the training of MSD personnel in the application of analytical and numerical methods to their individual problems.

2. Department 66-10 will continue to absorb the excess of computing and data reduction loads generated by other Lockheed divisions. Its Applied Mathematics Staff will be available to help solve difficult computational problems, to develop new simulation methods, to evaluate computing procedures used in accounting and to investigate the feasibility of further mechanization of Lockheed business problems. The Applied Mathematics Staff of Department 66-10 will be a source of mathematical consultant advice readily available to the entire corporation.

3. The extra capacity of Department 66-10 required for peak loads will be applied to outside business. This policy will avoid hasty expansion and minimize overtime following the award of large contracts. Outside business serves to smooth out fluctuations in internal demand in Department 66-10, to diversify MSD activities and to make profit for the company.

C. The reputation of Department 66-10 will be based on its performance and not on its size or jurisdiction. To have satisfied customers is one of the most important goals for Department 66-10.

D. For its survival, Department 66-10 must be able to recruit the ablest, best prepared and most dedicated young men and women into its service. To attract and hold such people, the promise of a career or livelihood or of economic success is not enough. The Department must also give them esprit de corps.

PX 71900-9-(9:23)

II. GENERAL PRINCIPLES AND OPERATING PROCEDURES

A. Everybody should have only one supervisor (except technical supervision on short projects).

B. There should be reasonable opportunities for advancement.

C.

1 All members of the department are encouraged to write technical articles for publication by MSD or in scientific journals.

2. It is in the interest of the department that members attend scientific meetings to further their own specialized knowledge in the interest of MSD and to learn about new developments elsewhere. The number of employees from MSD attending any one given scientific meeting is restricted by MSD policy.

3. It is in the interest of the department that its members try to further their own education and obtain advanced degrees.

D. Since frequent changes of equipment for Department 66-10 will be necessary, the department should be organized according to functions and not around existing or proposed equipment; thus, no reorganization will be required when such equipment changes are made.

E. Decisions should be made at the lowest possible level by supervisors of the smallest organizational units solely affected by the decision.

F. The department should be the lowest unit for administrative control from above. Within the department, the work should be broken down along functional lines.

G. Each organizational group should be provided with information it needs to measure its own performance. Each section should have the means of improving its own production-line activities.

H. Internal records and controls should be minimized. Only records which are of value should be kept. Internal controls should cost less than what they are supposed to save.

I. The work of the department should be divided into different levels. Personnel with the highest technical qualifications should work only at the highest level. For example, in general, a Ph.D. mathematician should not code problems and an electronic engineer should not perform routine maintenance. However, in emergencies, it may be necessary for even highly qualified technical personnel temporarily to perform duties which should, under normal circumstances, be assigned to personnel with less technical background or experience.

J. Prospective customers should be bothered as little as possible with paper work when they want Department 66-10 to help them solve one of their problems. Beyond stating explicitly what he wants Department 66-10 to do for him, a prospective customer should only need his own department's approval of a Department 66-10 estimate, but he should not be asked to participate in any more paper work. However, a written problem statement by the prospective sponsor is sufficient authorization to start the problem immediately on an interim basis until the administrative problems have been resolved.

K. During the move to the Bay Area, the department may be forced to depart temporarily from some of the general principles and operating procedures it would like to follow.

III. MAIN FUNCTIONS OF DEPARTMENT 66-10

A. Obtain analytical or numerical solutions to problems as a service function. It is usually not sufficient that an answer to a problem is found, but it must be furnished on time. A considerable amount of mathematical effort by the department may be involved in the formulation of mathematical models of the selection of numerical procedures or the error analysis of some of these problems.

B. Reduce flight test and other experimental data, participate in the pre-flight systems checkout and the analysis of the reduced data.

C. Operate and, if necessary, maintain the computing equipment assigned to Department 66-10.

D.

1. Provide a mathematical consulting service by loaning members of the Applied Mathematics Staff to projects requiring the services of high-level mathematicians.

2. Provide a consulting service on new applications of digital and analog computers both for the mechanization of scientific engineering and business problems and for the simulation of complex systems.

3. Provide a consulting service to projects and other departments on data processing problems regarding instrumentation and data acquisition requirements.

E. Conduct original research as a supporting function and directed toward possible application to the solution of scientific, engineering and business problems confronting Department 66-10, MSD or the Lockheed Corporation.

F. Make recommendations.

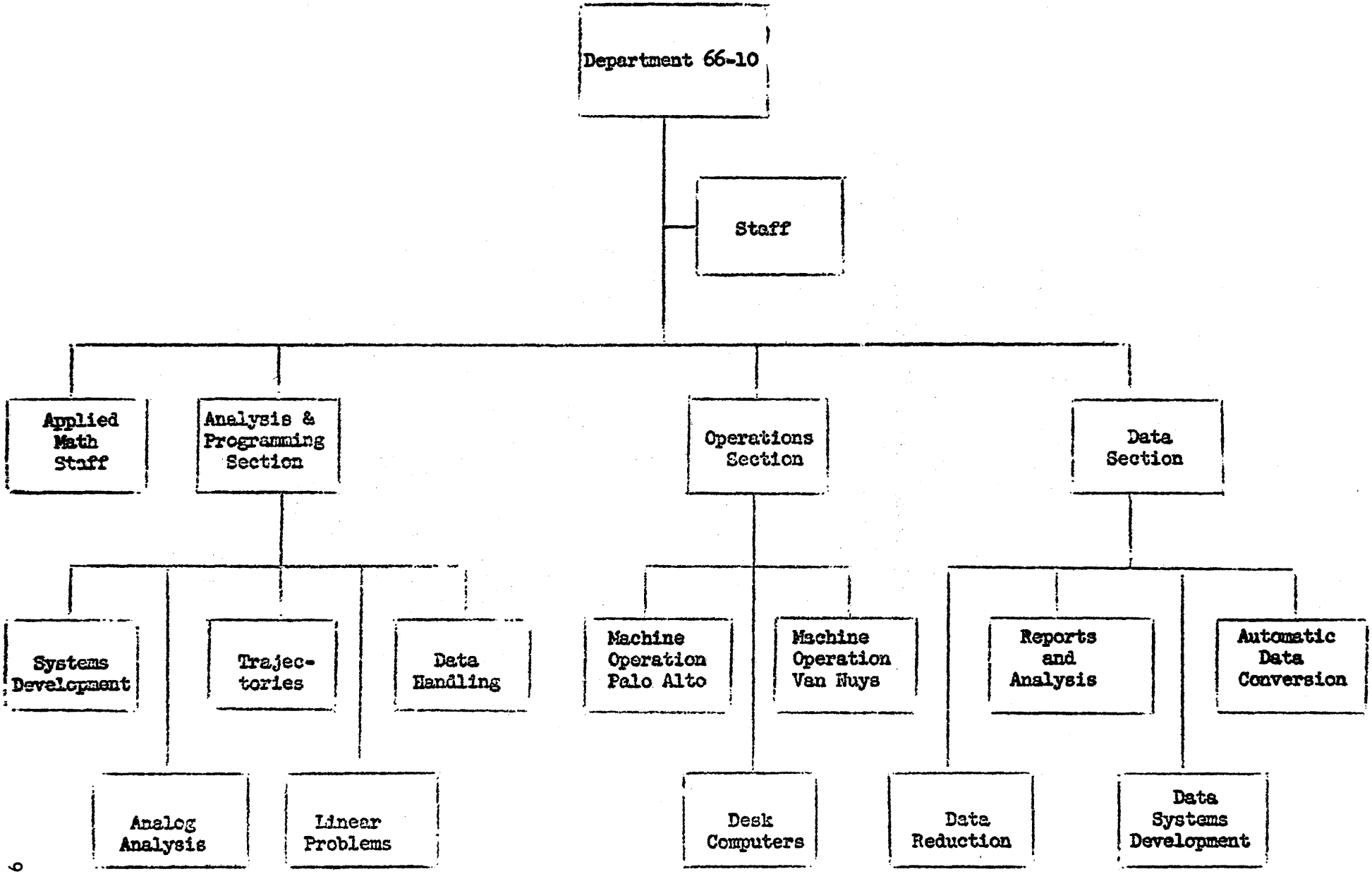
1. To management on policy matters affecting Department 66-10.

2. To management for new or improved computing and data reduction equipment.

G. Provide the services of a pool of desk computers on a loan basis to short-term projects.

H. Keep up to date on new developments in mathematics and the fields of computing, instrumentation and data processing.

IV. ORGANIZATION CHART



V. FUNCTIONS OF INDIVIDUAL UNITS

A. The head of the department exercises general administrative and technical supervision.

B.

1. The Analysis and Programming Section is responsible for the programming and coding of all problems solved on the department's general purpose digital and analog computers. In addition, it is responsible for the solution of all computational problems sent to Department 66-10 with the exception of data reduction problems.

2. a. The Systems Development Group is responsible for setting up and improving coding procedures and for improving the operation of the Analysis and Programming Section. It does not take part in the solution of outside problems sent to Department 66-10.

   b. Through the Systems Development Group, the Analysis and Programming Section is also responsible for keeping up to date on new cooperative coding efforts, such as PACT, SHARE, USE and others and for contributing to them as feasible. It should maintain liaison with programming groups at other installations and maintain up-to-date descriptions of different coding procedures and a library of subroutines.

3. The four programming groups should have competence primarily in the solution of problems suitable for analog computation, trajectory problems, linear problems and data-handling problems. The Data Section has first priority on the services of the data-handling programming group.

4. The Analog Analysis Group performs the following functions:

   a. Program problems for the analog computers.

   b. Operate and maintain the analog computers assigned to Department 66-10.

   c. Maintain general supervision over physical equipment used in conjunction with the analog computers of Department 66-10.

   d. Maintain state-of-the-art knowledge and make recommendations on future facilities and equipment requirements.

   e. Cooperate in the acquisition, operation and maintenance of any analog-to-digital and digital-to-analog conversion equipment when such equipment is intended for use with the analog computer.

5. Promising members of the programming groups should have an opportunity to contribute both to the work of the Systems Development Group and to the mathematical and numerical analysis of problems which would usually be carried out by members of the Applied Mathematics Staff. Each group leader should maintain close liaison with the Applied Mathematics Staff and the Systems Development Group and, whenever possible, contribute to either one of these two activities.

C. The Data Section is responsible for the following functions:

1. The processing of data as requested.

2. Participation in instrumentation checkout operations and evaluation of the final laboratory calibrations.

3. Production of a summary report describing the results of systems checkout runs conducted prior to shipment of a vehicle to the test site.

4. The publication of a post-flight conference "quick look" flight data report and conducting the post-flight conference, as well as editing minutes of the post-flight conference, including summary and conclusions drawn.

5. The publication of the final flight test data report (when required) covering all data reduced.

6. Analysis of telemeter performance to keep instrumentation personnel informed of the performance of their equipment, including accuracy determinations.

7. Act as consultants for future data acquisition requirements to all customers requiring this service.

8. The maintenance of the flight data file on all projects.

9. The operation and maintenance of all special purpose data-handling equipment assigned to the section.

10. Maintain state-of-the-ert knowledge and make recommendations on future facilities and equipment requirements.

In order to evaluate and improve its own operating procedures, the Data Section will have a special account supporting several employees in the Data Systems Development Group for this purpose on a full-time basis outside of all production activities.

E. The Operations Section is responsible for the following functions:

1. The scheduling, operation and, if necessary, the maintenance of the general purpose digital computing equipment assigned to Department 66-10.

2. The maintaining of machine records and the preparation of machine performance and usage reports.

3. Supervision of the machine areas as well as of key punching and desk computer services both within the department and for outside sponsors.

4. Coordination of the recommendations as to detailed specifications for new general purpose digital computing equipment based on future needs.

5. Acceptance tests on new digital equipment.

F. Members of the Applied Mathematics Staff, in general, spend at least half of their time on problems sent to Department 66-10 for solution on a service basis. The remaining time should be divided between the following functions:

1. Research in their own chosen field with proper approval. This type of research work must be reasonably close to MSD applications and funds have to be available from the MSD general research fund for this purpose.

2. It keeps up to date on new numerical methods and mathematical procedures on digital or analog computers. The securing of results from research in mathematical models for physical, data reduction and business problems as well as from numerical methods and error analysis for both digital and analog computers is included in this function.

3. Liaison with mathematical analysis groups at other computing centers.

4. Members of the Applied Mathematics Staff may be loaned to organizations outside of Department 66-10 to work on specific problems for a specified length of time.

G.

1. Even though the Applied Mathematics Staff does, in general, not require any technical supervision, its size does make it imperative to have a representative who serves as a senior staff member for the Applied Mathematics Staff.

2. Members of the Applied Mathematics Staff are available for consultation on technical problems by any member of Department 66-10 without any formalities.

DRUM ASSIGNMENTS
REVISED 06-01-56

UTILITY ROUTINE LIBRARY

DRUM ASSIGNMENTS

40000 GROUP ENTRANCES

	DECIMAL	OCTAL
PROGRAM START	40000	40000
FRI-0	40001	40001
MII-0 INSPECT	40002	40002
MII-0 INSERT	40003	40003
MDP-0	40004	40004
MDP-1 TAPE	40005	40005
STT-0	40006	40006
TST-0	40007	40007
CMP-0	40008	40010
	40009	40011
FPP-0 SNAP	40010	40012
FPP-0 SNIP	40011	40013
SAM-0	40012	40014
MDP-4	40013	40015
MDP-1 CARDS	40014	40016
CRI-1	40015	40017
COMMON EXIT	40016	40020
COMMON EXIT	40017	40021
RPH-0 READ	40018	40022
RPH-0 PUNCH	40019	40023
	40020	40024
	40021	40025
	40022	40026
	40023	40027
DEMONSTRATION ROUTINE	40024	40030
DEMONSTRATION ROUTINE	40025	40031
DEMONSTRATION ROUTINE	40026	40032
	40027	40033
	40028	40034
	40029	40035
LIBRARY TRANSFER TO M.T.	40030	40036
MDP-3	40031	40037
RESTORE ES	40032	40040

PX 71900-9-(9:24)

THE RAMO-WOOLDRIDGE CORPORATION
LOS ANGELES 45, CALIFORNIA

UTILITY ROUTINE LIBRARY

TABLE OF CONTENTS

MDP-0	12-09-55	THE FLEXOWRITER MEMORY DUMP, REVISED
MDP-1	12-09-55	THE BIOCTAL MEMORY DUMP, REVISED
MDP-2	12-09-55	THE OCTAL CARD DUMP
MDP-3	12-09-55	CHANGED WORD POST MORTEM
MDP-4	05-01-56	OCTAL CARD DUMP
MII-0	05-01-56	MANUAL INSPECTION AND INSERTION
MTI-0	11-30-55	LINEAR MATRIX EQUATION SOLVER
NRT-0	12-01-55	NTH ROOT ROUTINE
NUI-3	05-01-56	NUMERICAL INTEGRATION BY THE GILL METHOD
NUI-4	05-10-56	FLOATING POINT GILL METHOD
RAN-0	05-20-56	NORMALLY DISTRIBUTED PSEUDO RANDOM NUMBERS
RPH-0	05-23-56	COLUMN HEADING ROUTINE
SAM-0	08-09-55	AUTOMATIC SAMPLER
SIN-0	05-01-56	CENTRAL EXCHANGE SINE-COSINE ROUTINE
SIN-1	05-01-56	POLYNOMIAL MULTIPLY SINE-COSINE ROUTINE
SIN-2	05-01-56	SMALL ANGLE SINE-COSINE ROUTINE
SIN-3	08-10-55	FLOATING POINT SINE-COSINE ROUTINE
SIN-4	05-15-56	FLOATING POINT SINE-COSINE
SNI-1	09-12-55	ARCSINE-ARCOSINE ROUTINE
SNI-2	09-12-55	FLOATING POINT ARCSINE-ARCOSINE ROUTINE
SQR-0	05-01-56	SQUARE ROOT ROUTINE
STT-0	12-09-55	STORAGE TO MAGNETIC TAPE TRANSFER
TNI-0	05-01-56	ARCTANGENT ROUTINE
TNI-1	08-10-55	FLOATING POINT ARCTANGENT ROUTINE
TST-0	12-09-55	MAGNETIC TAPE TO STORAGE TRANSFER
URT-1	10-03-55	UTILITY ROUTINE TRANSFER - MAGNETIC TAPE TO DRUM
URT-3	08-23-55	UTILITY ROUTINE TRANSFER - DRUM TO MAGNETIC TAPE

CUMULATIVE ERRATA

Programming and Operating Conventions

Paragraphs 6, 8, and 9 should be deleted.

CMP-0 10/ /55

Pages 8A and 10 dated 4/18/56 should be included.

Page 8 - delete from "Subroutines" to end of page.

Page 9 - delete first three lines.

Page 9, paragraph two, and page 10, paragraph numbered 4 - should read "prints 'CMP-0'" and not "prints 'check sum fails'".

CVF-0 11/14/55

Page 5, line 12 from the bottom should read "n = (50000b)" and not "n + 1 = (50000b)".

DIE-0 7/26/55

Page 4, last line should read: $b = (\frac{\pi}{2}) \cdot n \cdot 2^{-4}$ and not $b = (\frac{\pi}{2}) \cdot n \cdot 2^{4}$

EGN-0 5/1/56

Page 8, line 14 should start "or the ERA paper tape reader)....."

Page 16 line 7 should start "if $n \leq 38$,.....

EXP-3 8/10/55

Page 1, drum assignment should read "63766 b through 64044 b."

FPP-0 5/25/56

Should include SNAP page 6 dated 5/10/56.

SNAP, page 5, last sentence should read: "If it is desired to read in less than four numbers per card, then the associated address field of the decimal number to be ignored must be left blank".

SNAP, page 6, first line below table should start "If the exponent or the mantissa....."

SNAP Smapler trace, page 1, the last sentence of paragraph b should be replaced by: "Restoring the library from magnetic tape loads an all zero word into cell 71777b. If this word is not changed, a complete trace of all SNAP commands is automatically performed."

FRI-0 12/9/55

Page 4, following paragraph numbered 5, add "Note: If there is a sixth level punch in the second of two consecutive frames having seventh level punches the stop is bypassed. The check sum is cleared and the reading continues. This will still be an illegal combination which will halt an ERA photo-electric reader".

HTO-0 7/25/55

Page 1, drum assignment should read "62504 b through 63037 b".

MDP-2 12/9/55

Page 1, opposite TYPE add "obsolete. available on symbolic cards"

Page 2, first sentence below the list of card column assignments should read: "Any card is omitted if each of the four words to be punched consists of 36 zeros or 36 ones, and in this event the next card produced carries a punch in the 12 row of column 9".

MDP-3 12/9/55

Page 2, line 12, reference to MDP-2 should read MDP-4.

Page 2, line 14, should start "each card contains six cards".

SNI-1 9/12/55

Page 2, under programming instructions add: "The ranges of the results are the principal values, defined as follows:

$$-\pi/2 \leq \text{arcsine } \theta \leq \pi/2$$

$$0 \leq \text{arcosine } \theta \leq \pi \text{ "}$$

PX 71900-9-(9:25)

SNI-2 9/12/55

Page 2, under programming instructions add: "The ranges of the results are the principal values, defined as follows:

$-\pi/2 \leq \text{arcsine } \theta \leq \pi/2$

$0 \leq \text{arcosine } \theta \leq \pi$ "

TNI-1 8/10/55

Page 2, under programming instructions add: "The range of the result is the principal value, defined as follows:

$-\pi/2 \leq \text{arctangent } x \leq \pi/2$ "

URT-1 10/3/55

Page 2 following paragraph six add "ES is cleared".

PSEUDO-RANDOM NUMBER GENERATOR SUBROUTINE

by - Harold Dahlbeck

The basic method used in this subroutine was described by D. H. Lehmer in The Annals of the Computation Laboratory of Harvard University, Volume XXVI, Proceedings of a Second Symposium of Large-Scale Digital Calculating Machinery Harvard University Press, Cambridge, Massachusetts, 1951.

The method itself requires the following simple form to be used as the iteration formula:

$$X_{i+1} = CX_i \pmod{p} \qquad i = 1, 2, 3, - - -$$

Although this method is considered to be quite optimal in guaranteeing both maximum randomness and greatest periodicity of the X_i, there are considerable problems involved in the obtaining of c and p subject to the following conditions:

1. p should be the largest prime number which can be contained in a register.
2. c must be of the form R^1 when R is required to be a primitive root of p and 1 has to be relatively prime to p-1. In addition c should be as large as possible.

The following two numbers satisfy the above conditions:

$p = 2^{35} - 31 = 34, 359, 738, 337$

$c = 5^{13} = 1, 220, 703, 125$

A subroutine for generating the X_i is given below. The periodicity of the X_i will be $p - 1 = 2^{32} - 32 = 34, 359, 738, 336$. Any positive number may be used as a starting point for this series.

PX 71900-9-(9:25)

1103 Subroutine for Pseudo-Random Numbers

01000	71	01777	01776	$5^{13}, R_1 \longrightarrow A$
01001	73	01775	10000	$(5^{13}.R_1) \bmod p \longrightarrow A$
01002	11	20000	01777	$A \bmod p \longrightarrow R_1$
01775	37	77777	77741	$2^{35} - 31$
01776	01	10604	71625	5^{13}
01777	00	00000	00001	R_i

A Linear Programming Routine for the 1103 Computer

General Remarks:

A working draft of a linear programming routine for the 1103 is now in operation. The routine finds a basic set of values of the variables in a linear form, which will maximize or minimize the value of that linear form, (hereafter called "profit function"), subject to a set of linear restraints. These restraints may be equations or inequalities, and are acceptable with or without slack variables. However, if slack variables are included, they should be clearly labelled as such; if they are omitted, the sense of the inequalities must be indicated.

The method followed is the Alternate Algorithm of the Revised Simplex Method of Dantzig, as described in Rand Corporation manual RM-1268, with certain modifications. Since zero suppression is used throughout, the size of the problem which can be handled by this program will depend upon the number of zeros in the original matrix of coefficients and in the columns stored for the inverse matrix in product form. In any case, the number of restraints cannot exceed 106, the number of variables, including slack variables, must be less than 258, and the product of the two dimensions should not exceed about 15,000.

The time required for solving a particular system depends upon the size of the matrix, the number of non-zero elements, and the number of iterative cycles performed in reaching a solution. In general, the time will range from a few minutes for a small system to about 2½ hours for the largest system solvable by this program. High speed has been achieved by confining the cyclic program and vector storage to the rapid-access memory and the magnetic drum (hence the limitation on size of the system). Magnetic tape units supply programs for input, computation, and output, and receive intermittent dumps of the entire contents of rapid-access storage and drum as protection against unforseen interruption, and to provide for output of results of previous cycles. The program may of course be modified to handle larger systems by using magnetic tape storage during cyclic computations; however, this will considerably increase production time on the 1103. Some actual computation times are as follows:

Number of Restraints	Number of Variables Including Slack	Number of Iterations	Computation Time	Total Time, Incl. Input and Output
24	49	21	2 minutes	4.5 Minutes
26	37	33	3.3 "	6.2 "
27	59	31	4 "	6.5 "
63	88	59	11.7 "	15.6 "

PX 71900-9-(9:26)

The present program calls for input data on punched paper tape in Flexowriter code. The matrix is read in by rows just as they appear on the matrix sheet, with the profit function as the first row. A detailed description of input format is attached. While the paper tape is being read into the computer, the input program converts the decimal numbers to binary, assigns a slack variable of appropriate sign to each inequation and scales the elements of each row according to the numerically largest coefficient in that row. The matrix is then transposed, and each column is

assigned a scaling based on the numerically largest element in that column. The program then packs each column vector by retaining only its non-zero elements in sequence, preceded by three code-words displaying the pattern of zero and non-zero elements in the column. Packed columns are then stored in sequence on the drum, and a directory is prepared which records the starting address and scaling of each column vector.

Artificial variables are assumed where needed, but their columns are not stored, and should not be included in the matrix. The input program prepares the redundant equation which serves to eliminate the artificial variables during the first phase of computation.

Computation:

Floating vector arithmetic is used for addition and scalar multiplication of two vectors and for multiplying a vector by a constant. This greatly reduces computation time as compared with floating point arithmetic, yet avoids the overflow problems and loss of significance which arise in fixed point arithmetic. Significance retained in final answers will depend, of course, upon the amount of computation necessary to reach the answers; results to date have been excellent, with all answers correct to at least five significant digits.

The computation proceeds through two phases: Phase I eliminates artificial variables by maximizing the artificial variable in the redundant equation. When feasible solutions exist, computation then proceeds into phase II, which maximizes (or minimizes) the profit function itself. Details of the method may be found in the Rand Corporation literature.

Output:

The type of output will vary with the nature of the conclusions reached. In most cases, an optimum feasible solution will result; the values and identifying indices of the basic variables will be printed, as will the shadow prices, the back solution, and the quantities labelled delta-sub-j in the notation of RM-1268.

The inverse matrix itself is stored in the convenient product form. Thus, at the end of the computational program, the inverse matrix as it would appear at the end of any specified cycle is still available. Except for the inverse matrix, results of previous cycles are not retained. If these are desired, the program provides for output after specified cycles. It is recommended that output be kept to a reasonable minimum, to save computer time.

In case an infinite solution is indicated, the output program will identify at least one variable which is unbounded.

If the original system contains incompatible restraints, computation will halt when this is recognized through an unfeasible maximum or minimum. At least one restraint will be identified by the presence of a non-zero artificial variable as being inconsistent with others in the system.

In case of linear dependence among restraints, the program will proceed exactly as in the case of an optimum feasible solution, except that one or more artificial variables will be listed in the final basic solution, with assigned values of zero. These serve to identify redundant restraints which may then be deleted by the analyst,

though they do no harm other than to increase computation time.

Programmer's Comments:

The linear program described above is presented as a working draft, rather than as a completed product. It will be expanded and adapted to the needs of customers as we become more familiar with these needs. As the result of experience gained in checking out and using the program, certain possibilities for improvement have become evident and will be incorporated in the program as soon as possible. One specific change contemplated in the present 1103 version is the addition of the facility for finding alternate solutions by imposing increments upon specified quantities and continuing the computational cycle. This is recognized as highly desirable and will be added as soon as possible.

Suggestions as to other services which might be incorporated in the linear program will be appreciated. We will of course be happy to answer any questions which might arise concerning the program.

Use of the Program on the 1103A:

The 1103 program is available in a slightly altered form for use on the 1103A. The only changes are in the addressing of the accumulator and quotient register, and in the references to magnetic tape; only 1024 words of core memory are used. Thus, running time on the 1103A with this program will be approximately the same as on the 1103. This time is governed to a large extent by the number of references to the magnetic drum. At present, the entire cyclic program, the floating vector subroutines, and the directory which lists vector storage locations, are kept on the drum because of limited space in the 1024 word rapid-access storage of the 1103. Portions of the program are then transferred from the drum to the core storage as needed.

A complete rewriting of the Linear Programming Routine for the 1103A has been initiated. Since the 4096-word core memory provides space for everything except the vectors themselves, at least half of the drum references will be eliminated, which in turn will reduce computation times by an estimated 40 or 45 percent.

	IC004				EXPLANATION
70765	01033	73	01247	10000	R = REMAINDER ⟶ (A)
70766	01034	11	20000	01266	STORE R
70767	01035	36	01213	01267	STORE R-1
70770	01036	35	01242	20000	R + 4 ⟶ (A)
70771	01037	73	01242	01270	STORE R + 4 DIVIDED BY 5
70772	01040	23	01270	01213	STORE R + 4 DIVIDED BY 5 MINUS 1
70773	01041	31	01266	00020	STORE
70774	01042	11	20000	01271	2R X 2^{15}
70775	01043	31	01266	00001	STORE
70776	01044	36	01213	01272	2R-1
70777	01045	45	00000	01047	
71000	01046	00	00000	00140	
71001	01047	37	00540	00540	POSITION
71002	01050	20	00000	0000	CARD
71003	01051	37	00540	00540	TO
71004	01052	20	00000	0000	PUNCH
71005	01053	15	01000	01077	SET INITIAL
71006	01054	15	01000	01107	DATA ADDRESS
71007	01055	11	00777	01264	SET COLUMN INDEX = M
71010	01056	11	01250	10000	MASK ⟶ (Q)
71011	01057	53	01251	01076	SET BLOCK TRANSFER
71012	01060	11	00537	01207	STORE 310 OCTAL
71013	01061	11	01260	20000	
71014	01062	47	01072	01063	INDEX = 0?
71015	01063	11	01271	20000	YES:
71016	01064	47	01065	01072	R = 0?
71017	01065	11	01250	10000	NO: MASK ⟶ (Q)
71020	01066	53	01271	01076	SET BLOCK TRANSFER
71021	01067	11	01270	01254	SET CARD INDEX
71022	01070	75	10002	01075	SET
71023	01071	11	01267	01252	NUMBER INDICES
71024	01072	11	01246	01252	NO: SET BLOCK BY COL. TRANSFER INDEX

PX 71900-9-(39)