DECUS
PROGRAM LIBRARY

DECUS NO.	8-652
TITLE	REGRESSION ANALYSIS PACKAGE
AUTHOR	Theodore E. Bridge
COMPANY	54 Williamsburg Drive Springfield, Massachusetts
DATE	September 2, 1973
SOURCE LANGUAGE	PAL

DECUS Program Library Write-up DECUS NO. 8-652

REGRESSION ANALYSIS PACKAGE

FOR THE PDP 8 WITH 4K OF CORE AND AN ASR33 CONSOLE

In Numerical Analysis , we have a therom stating that any curve may be represented by a polynomial equation ; if, you have enough terms in the equation. In actual practice, we can fit most curves with only a few terms . In this program, we assume that a dependent variable W may be represented by a polynomial function of independent variables (X, Y, Z). We enter data for many points , and ask the computer to find the coefficients that will give a least squares fit . We specify the number of terms in the equation . This we call THE DEGREES OF FREEDOM.

This package is a group of programs for making a multiple regression with up to 3 independent variables having up to 28 degrees of freedom. They are useful in two ways:

1. If you want the computer to find a formula that will calculate data given by a family of curves . For example, you might want to run a heat balance on a boiler . You would want the computer to determine the enthalpy of steam for many different values of temperature and pressure . Also, you might want to read the guarantee curves supplied by the boiler manufacturer.

2. You might want to find out if there is a pattern in a large mass of test data . These regression programs will find the pattern (if it exists) and will print the standard deviation from the pattern.

These regression programs are delivered on four tapes.

RP 5000-6777 — The first is the RESIDENT PACKAGE that we keep in core at all times for use also on other programs . It includes the 3 Page -- 27 Bit Floating Point Package DECUS 8-375B.

RA1 7000-7577 — The second, -- REGRESSION ANALYSIS part 1 -- calculates the regression coefficients.

RA2 4400-4777 — The third -- REGRESSION ANALYSIS part 2 -- will reread the data and expand the polynomial equations using the regression coefficients of part 1 . The calculated value of the dependent variable V is compared with the given value of W .

QD&QL 7600-7755 — The fourth -- QUICK DUMP and QUICK LOAD--- are in RIM format as they overwrite the BIN loader . Since the regression parameters must be loaded in a relocatable format , the BIN loader could not be used.

On the next page, we explain the regression formulas . We follow with a list of the programs in the order that they appear in core. We will discuss them in this order.

REGRESSION EQUATIONS

WE WILL SIMPLIFY THIS EXPLANATION BY USING A SPECIFIC FORMULA RATHER THAN A GENERAL ONE. WE WILL USE THE UP ARROW (↑) TO DENOTE EXPONENTIATION, AND THE PRIME (') TO DENOTE DIFFERENTIATION. V IS A POLYNOMIAL FUNCTION OF (X & Y). VARIABLES A, B, C, D, E, & F ARE THE COEFFICIENTS TO BE MULTIPLIED BY EACH SUCCESSIVE TERM.

$$V = A + BX + CX\uparrow 2 + DY + EYX + FYX\uparrow 2$$

IN THE ABOVE EQUATION, V HAS 3 DEGREES OF FREEDOM WITH RESPECT TO X (A PARABOLA) AND 2 DEGREES OF FREEDOM WITH RESPECT TO Y (A STRAIGHT LINE) . WE WANT TO FIND VALUES FOR A, B, C, D, E, & F THAT WILL GIVE A MINIMUM ERROR (SQUARED) WHEN SOLVED WITH ANY GIVEN VALUES FOR X AND Y. W WILL BE A GIVEN OR MEASURED VALUE OF THE DEPENDENT VARIABLE. V IS THE CALCULATED VALUE GIVEN BY THE ABOVE FORMULA.

THE ERROR SQUARED IS: $U = (V-W)\uparrow 2$

FOR A MINIMUM VALUE OF U , THE DERIVATIVE WITH RESPECT TO A WILL BE ZERO.

$U'/A' = 2\Sigma(V-W)\ V'/A'$, BUT $V'/A' = 1$; SO $\Sigma V = \Sigma W$

AND $\Sigma A + \Sigma BX + \Sigma CX\uparrow 2 + \Sigma DY + \Sigma EYX + \Sigma FYX\uparrow 2 = \Sigma W$

IN A LIKE MANNER, WE CAN DIFFERENTIATE WITH RESPECT TO B AND GET ANOTHER EQUATION :

$U'/B' = 2\Sigma(V-W)\ V'/B'$; BUT $V'/B' = X$; SO , $\Sigma VX = \Sigma WX$

$$\Sigma AX + \Sigma BX\uparrow 2 + \Sigma CX\uparrow 3 + \Sigma DYX + \Sigma EYX\uparrow 2 + \Sigma FYX\uparrow 3 = \Sigma WX$$

WE CAN CONTINUE THIS PROCESS AND GET 6 EQUATIONS WITH 6 UNKNOWN COEFFICIENTS. WE WILL ARRANGE THESE EQUATIONS IN MATRIX FORM.

REGRESSION MATRIX

N	$+\Sigma X$	$+\Sigma X\uparrow 2$	$+\Sigma Y$	$+\Sigma YX$	$+\Sigma YX\uparrow 2$	$=\Sigma W$
ΣX	$+\Sigma X\uparrow 2$	$+\Sigma X\uparrow 3$	$+\Sigma YX$	$+\Sigma YX\uparrow 2$	$+\Sigma YX\uparrow 3$	$=\Sigma WX$
$\Sigma X\uparrow 2$	$+\Sigma X\uparrow 3$	$+\Sigma X\uparrow 4$	$+\Sigma YX\uparrow 2$	$+\Sigma YX\uparrow 3$	$+\Sigma YX\uparrow 4$	$=\Sigma WX\uparrow 2$
ΣY	$+\Sigma YX$	$+\Sigma YX\uparrow 2$	$+\Sigma Y\uparrow 2$	$+\Sigma Y\uparrow 2X$	$+\Sigma Y\uparrow 2X\uparrow 2$	$=\Sigma WY$
ΣYX	$+\Sigma YX\uparrow 2$	$+\Sigma YX\uparrow 3$	$+\Sigma Y\uparrow 2X$	$+\Sigma Y\uparrow 2X\uparrow 2$	$+\Sigma Y\uparrow 2X\uparrow 3$	$=\Sigma WYX$
$\Sigma YX\uparrow 2$	$+\Sigma Y\uparrow X\uparrow 3$	$+\Sigma YX\uparrow 4$	$+\Sigma Y\uparrow 2X\uparrow 2$	$+\Sigma Y\uparrow 2X\uparrow 3$	$+\Sigma Y\uparrow 2X\uparrow 4$	$=\Sigma WYX\uparrow 2$

LIST OF PROGRAMS

4400--4450 DUMP REGRESSION COEFFICIENTS to tape.
S-4420
DRC

4451--4577 REGRESSION ANALYSIS part 2. This program rereads
S-4472 the data and calculates the error for each data
RA2 point. It prints the RMS error or the STANDARD
DEVIATION at the end.

4600--4777 POLYNOMIAL EXPANSION. This routine will calculate
JMS I A4600 the dependent variable, by the polynomial expansion
PLY of the regression coefficients.

5000--5021 MESSAGE. This routine may be used to type a
JMS I A5000 message that was loaded in core by QL at the
MES address in the AC.

5041--5177 CARRIAGE RETURN, PRINT OCTAL ACCUMULATOR, AND
CR, POA, FLOATING POINT TRACE. These are utility programs.
FPT

5200--5227 POOR MAN'S EDITOR to copy a tape with some control
S-5200 over the typewriter.
PME

5233--5337 ANTILOG. This program will calculate the anti-
JMS I A5271 logarithm of the Naperial logarithm given in the
ANTL floating point accumulator.

5400--6577 Five page FLOATING POINT PACKAGE with floating
JMS I A5400 output -- DECUS 8-375B.
FPNT

6502--6565 READ INTEGER. This program reads integer numbers
JMS I A6502 from the keyboard.
RINT

6600--6660 FLOATING INPUT routine to read floating point
JMS I A6600 numbers from the keyboard.
FIN

6661--6777 Routine to calculate the Naperian LOGARITHM of
JMS I A6661 the number in the floating point accumulator.
LOG

7000--7177 SOLVE SIMULTANEOUS EQUATIONS. This routine will
JMS I A7012 solve up to 28 simultaneous equations.
SSE

7200--7577 REGRESSION ANALYSIS part 1. This program builds
S-7342 the regression matrix, and calls SSE to solve
RA1 for the regression coefficients.

LIST OF PROGRAMS continued

7600--7675 QUICK DUMP. This program will dump one page of
S-7600 core starting at the address in the switch register.
QD It will continue on the next page every time you depress CONT after each halt. Two characters will be typed to define each word. Each line will start with a bell (207) followed by the address at which the line starts in core.

7700--7753 QUICK LOAD. This program will load a tape cut by
S-7700 QD or by DRC. You must type the loading address
QL before loading a tape cut by DRC. (if you forget, you may clobber core.)

The two preceeding programs are on a tape in RIM format for loading by the RIM loader at 7756.

4400-4450 DRC - DUMP REGRESSION COEFFICIENTS --S-4400

This program is automatically entered on exit from RA2. A halt at the beginning of the program will give you time to turn the punch on. If you know where the coefficients are to be stored, you should type BELL (207) ADDR before you depress CONT. This will tell the loader QL where to store the coefficents. You should also set some identifying pattern in the switch register. This pattern will be stored in N4 where it may later be examined to identify the data. All other parameters are set by the program from data entered in RA1.The following parameters are put on tape.

NL = a parameter to tell PLY whether or not to take the logarithm of a variable before calculating the dependent variable.

	NL = 0	1	2	3	4
Take antilog of W ?	YES	YES	YES	YES	NO
Take log of X ?	YES	YES	YES	NO	NO
Take log of Y ?	YES	YES	NO	NO	NO
Take log of Z ?	YES	NO	NO	NO	NO

N1 = Negative number of degrees of freedom for variable X .
N2 = Negative number of degrees of freedom for variable Y .
N3 = Negative number of degrees of freedom for variable Z .
N4 = Identification. Arbitrary pattern set in switch register.
N5 = Number of words required to store the regression coefficients.

The regression coefficients follow.

4451-4577 -RA2 -- REGRESSION ANALYSIS part 2 -- S-4472

This program will reread all of the input data. For each point, a value V will be calculated using the regression equation with the coefficients determined in part 1.See page 2 . The error (V-W) will be determined. For each point, the following values will be printed as shown:

V	(V-W)	POINT NO.	
W	X	Y	Z

At the end of the run, the RMS error will be typed. This program may be entered by depressing CONT after the halt at the end of part 1 at 7264. However, if you specified more than 26 degrees of freedom, this program will have been overlayed by the matrix built in part 1. So, you must reload this program and start at 4472.

4600-4777 PLY POLYNOMIAL EXPANSION -- JMS I A4600

This routine will expand the polynomial equation. On return, the floating point accumulator will hold the calculated value V. You will need this routine in your final program that is to read the data correlated by the regression coefficients. Use the following sequence of instructions in your program.

```
ANL,      NL-1          / ADDRESS OF  NL-1
A4600,    4600          / ADDRESS OF  PLY
AOX,      X             / ADDRESS OF  X

          CLA
          TAD  ANL      / GET ADDRESS OF  NL-1
          JMS I A4600
          AOX           / ADDRESS OF  X
                        / CONTINUE WITH PROGRAM.
                        / FAC WILL CONTAIN  V

X,        0000          / INDEPENDENT VARIABLES
          0000          / LOCATED FROM THIS POINT ON.
          0000
Y,        0000
          0000
          0000
Z,        0000
          0000
          0000

NL,       0000          / REGRESSION PARAMETERS FOLLOWED
N1,       0000          / BY REGRESSION COEFFICIENTS
N2,       0000          / LOCATED FROM HERE ON IN ORDER.
N3,       0000          / QL  WOULD BE USED TO LOAD THE
                        / DRC  TAPE TO PLACE THEM HERE.
```

5000-5021 MES MESSAGE -- JMS I A5000

This routine may be used to print a message that was loaded in core by QL at the address given in the AC. If your message includes :

1. Back arrow up arrow (← ↑) it will generate a carriage return line feed when printing.

2. Two back arrows (← ←) will terminate the message

Each pair of symbols, given above, must be contained in one word. When using QUICK LOAD to load your message, the AC will contain 7777 if you are entering the first character of the pair. Otherwise, it will be blank. So, if the AC is blank when you want to enter the first back arrow (←), hit a space first.

5042-5105 CR CARRIAGE RETURN -- JMS I 74

This routine will issue a carriage return - line feed. The accumulator will be cleared. It will also keep a line count and type 6 blank lines whenever 60 lines have been counted. If you call this routine with 7777 in the AC, it will skip to the next page.

5106-5133 POA PRINT OCTAL ACCUMULATOR -- JMS I 73

This routine will cause the contents of the accumulator to be printed in octal.

6305-6312 TYPE -- JMS I 75

This routine will cause the character in the accumulator to be typed and the accumulator cleared. This routine furnished with DECUS 8-375B was changed as follows to improve timing.

```
6305  0000  TYPE, 0
6306  6041        TSF
6307  5306        JMP  .-1
6310  6046        TLS
6311  7200        CLA
6312  5705        JMP I TYPE
```

5134-5177 FPT FLOATING POINT TRACE -- S-5134

To use this trace, you must first establish the linkage. You can use ODT (Octal Debugging Technique) to insert codes as follows.

```
1.    Insert  5134  at address  0077
2.    Insert  5477  at address  5404
3.    Insert  LOW   at address  5050
4.    Insert  HIGH at  address  5051
```

LOW is the lowest address to be traced.
HIGH is the higest address to be traced.

All floating point interpretive instructions that fall in the range between LOW and HIGH will be traced. First, the carriage will be returned, then the address will be typed in octal. Finally, the FAC will be typed in E format.

LOW and HIGH will usually be on the same page. But, this is not necessary if they are both on the same side of address 4000. We cannot have one positive and one negative number used to specify the tracing range. On the next page, we show you two traces that were run on the ANTL routine and one on the LOG routine.

```
1620G   trace  e^(-.868)
-.868   A
5273 -1.252259E+00
5274 -1.252259E+00
5303  2.522591E-01
5304  2.522591E-01
5306  2.522591E-01
5307  1.000000E+00
5310  7.477409E-01
5311  7.477409E-01
5324  2.500000E-01
5325  7.477409E-01
5331  2.591472E-01
5332  2.591472E-01
5333  6.715728E-02
5334  6.015735E+01
5335  6.015735E+01
5336 -6.018043E+02
5337 -1.000384E+01
5340 -1.026298E+01
5341  1.752033E+00
5342  1.752033E+00
5343  2.591472E-01
5344  5.182944E-01
5345  2.958246E-01
5346  1.295825E+00
5347  1.295825E+00
5350  1.679162E+00
5351  4.197904E-01 4.197904E-01

2.00148  A     trace e^(2.00148)
5273  2.887525E+00
5274  2.887525E+00
5303 -8.875249E-01
5304 -8.875249E-01
5305 -8.875249E-01
5324  4.000000E+00
5325 -8.875249E-01
5331  3.075927E-01
5332  3.075927E-01
5333  9.461326E-02
5334  6.018480E+01
5335  6.018480E+01
5336 -6.018043E+02
5337 -9.999273E+00
5340 -1.030687E+01
5341  1.708151E+00
5342  1.708151E+00
5343  3.075927E-01
5344  6.151854E-01
5345  3.601469E-01
5346  1.360147E+00
5347  1.360147E+00
5350  1.850000E+00
5351  7.399999E+00 7.399999E+00
```

```
1620G    trace  log(64)
64  L
6663  6.400000E+01
6707  7.000000E+00
6710  6.400000E+01
6716  5.000000E-01
6717  1.207107E+00
6720  1.207107E+00
6721  5.000000E-01
6722 -2.071068E-01
6723 -1.715729E-01
6724 -1.715729E-01
6725  2.943725E-02
6726  2.943725E-02
6727  5.989786E-01
6730  1.763228E-02
6731  9.791028E-01
6732  2.882209E-02
6733  2.914213E+00
6734 -4.999999E-01
6735 -9.999999E-01
6736  6.000000E+00
6737  4.158883E+00 4.158883E+00
```

5200-5227 PME POOR MAN'S EDITOR -- S-5200

This is a short program that will copy a tape with some control over the typewriter.

1. If you set a character code in the switch register, the typewriter will halt before typing the selected character. If you want to type the selected character, depress CONT. If you want not to type the character, depress EXAM, CONT.

2. If you put SW0 up, every character will be selected; so, you can step accross the page -- one character at a time, each time you depress CONT.

3. If you want to add a few characters to the output tape that were not on the input tape, you can turn the reader off and depress CONT. All characters typed will appear on the output tape.

4. If you put SW3 up, it will temporarily cancel your character selection, and the typewriter will keep going. When you get near the point where you want to stop; put it down and the typewriter will halt before typing the selected character.

5234-5270 FIX -- JMS I A5234

This routine will shift the FAC left or right untill the decimal point registers at the low order end of AC1. AC2 and AC3 will be set to zero. Any digits shifted out of the high order side of AC1 will be lost and no error flagged. The routine will return with the integer portion of the number in the machine accumulator. If the sign was negative, it will be two's complimented.

5271-5377 ANTL ANTILOGARITHM -- JMS I A5271

On return from this routine, the floating point accumulator will be replaced with the antilogarithm (e^x) of the Naperian logarithm that was in the FAC when called.

6502-6565 RINT READ INTEGER -- JMS I A6502

This routine is used by FIN below to read in the decimal digits and convert to binary. Therefore, it uses the conventions described below. It will load the number in AC3 overflowing into AC2, and thence into AC1. If you keep your integers smaller than 2048 , you can use single precision. You should test SGN at addr. 51 and compliment if it is negative.

6600-6660 FIN FLOATING INPUT -- JMS I A6600

This routine will read decimal digits from the keyboard, convert to binary, and load the number in the FAC. E FORMAT IS NOT AN ACCEPTABLE INPUT FORM.

Too many digits at input will produce an unflagged error.

We cannot handle 11 digits.

```
34567890120.    I 2.081518E+08  Unflagged error
```

unless the first digit is 2 or less

```
23456789010.    I 2.345679E+10
29876543210     I 2.987654E+10
```

Leading zeros don't count

```
00000987654321 I 9.876543E+08
.9876543212345 I 1.529830E-03 Unflagged error
.0000098765432 I 9.876542E-06
```

The sign may appear at either end of the number.

-123.456 is the same as 123.456-

Other than minus sign (-) and decimal point (.) , all characters coded less than (260) are ignored. Carriage return (215), line feed (212), plus sign (253), and space (240) are all ignored by the input routine.

Any character coded greater than nine (271) will terminate the input. The terminating character will be saved in address (47). Your program should examine this register to identify whether and where the number is to be stored. Thus, you can arrange to have incorrect numbers overwritten by corrected ones.

6661-6677 LOG LOGARITHM --JMS I A6661

On return from this routine, the floating point accumulator FAC will be replaced with the Naperian logarithm of the number that was in the accumulator when called. A halt at 6667 will occur if the FAC is zero or negative when called.

The logarithm and antilog routines give about six digit accuracy.

LOGARITHMS		ANTILOGARITHMS	
.00000987654321	L-1.152535E+01	-11.52535	A 9.876536E-06
23456789010.	L 2.387843E+01	23.87843	A 2.345683E+10
.4197904	L-8.679998E-01	-.868	A 4.197904E-01
7.4	L 2.001480E+00	2.00148	A 7.399999E+00
987654321	L 2.071084E+01	20.71084	A 9.876489E+08
98765432	L 1.840826E+01	18.40826	A 9.876541E+07

7000-7177 SSE SOLVE SIMULTANEOUS EQUATIONS -- JMS I A7102

This routine will solve a set of simultaneous equations that have previously been loaded in matrix form starring at address 236. The accumulator must be clear at entry. The single precision word at address 100 must be set equal to the negative number of equations. The word at address 105 must be a positive number equal to the number of equations, multiplied by 3 . It shows how many words are needed to store one column of the matrix. The unknowns (or answers) will be printed before exit. Also, the answers will be stored as floating point numbers (tripple precision) starting at address 106. The matrix of equations must be stored in column major order (as in Fortran) with the given constants to the right of the equal sign stored in the last column.

7200-7377 RA1 REGRESSION ANALYSIS part 1 S-7342

This program will first read four parameters --A, B, C, & D.

A = number of degrees of freedom for variable X
B = number of degrees of freedom for variable Y
C = number of degrees of freedom for variable Z
D will tell which variables are to be converted to logarithms before use.

	D =	0	1	2	3	4
Take log of	W ?	NO	YES	YES	YES	YES
Take log of	X ?	NO	NO	YES	YES	YES
Take log of	Y ?	NO	NO	NO	YES	YES
Take log of	Z ?	NO	NO	NO	NO	YES

Some data will correlate better if plotted on logarithmic or semilogarithmic graph paper. For this kind of data, you should set D at a value that will cause the logarithm to be taken.

These parameters may be read in any order. If you find that you have entered a wrong number, you may overwrite it with a corrected one. When you have all parameters correctly entered, enter an E to terminate the list.

To enter the above parameters, you simply type a number followed by the designating letter A, B, C, or D. Space, Carriage Return and Line Feed will be ignored. If not entered, A, B, & C will default at (1.) . D will default at zero.

Now that the parameters are entered, you may enter the dependent variable W; and the independent variables X, Y, & Z. For each data point, a value for W and one for X should be entered; with Y and Z optional.

Here also, you may overwrite any variable incorrectly entered with a corrected number. Variables may be entered in any order. You type the number, followed by the designating character W, X, Y, or Z. Space, Carriage Return , of Line Feed will be ignored. The data for one point is terminated with a left bracket ([ ). If data for a variable was not entered, the previous value will be used. Any character coded less than (327) will cause its associated number to be discarded. You may terminate the run with a back slash (\ ) .

The program assumes that the dependent variable W may be represented as a polynomial function of the independent variables X, Y, & Z .It will develop regression coefficients for use in the equation as explained on page 2 . It will develop coefficients to give a least squares error.

You must enter more data points than there are degrees of freedom. In other words, the number of data points must be more than the product (A x B x C). You ought to have many more. If you do not have enough points, you may halt on a divide by zero at addr. 6064.

The example in the appendix will help to explain the entry of data. You will probably want to cut a tape in the local mode for use in both part one and part two. You can use the POOR MAN'S EDITOR -- PME at 5200 to make minor corrections to your data tape.

This program calls REG to form the matrix, and SSE to solve for the regression coefficients.

7413-7577 REG REGRESSION MATRIX -- JMS I A7413

This routine will read the values for the dependent variable W and the independent variables (X, Y, & Z) and form the regression matrix as shown on page 2 . It will return to RA1 after the endfile character -- back slash (\ )-- is read .

The next two programs are on a tape in RIM format. They will overlay the BIN loader ; so they must be loaded by the boot at 7756. They are not really part of the REGRESSION ANALYSIS SYSTEM , but are needed to put the regression coefficients in core at relocatable addresses for use by other programs.

7600-7677 QD QUICK DUMP -- S-7600

You can use this program to dump or display any part of core memory at 2 characters per word. This is useful when you are debugging a program that changes its own instructions, and you want to find out where it went wrong. You take a listing of any page in core in about 30 seconds. You can compare it with a previous listing before it went wrong.

To start the dump:
1. Put 7600 in memory address register.
2. Set the first address to be displayed in the switch reg.
3. Depress CONT.

You can use the table on the next page to translate any pair of characters in the listing into the equivalent octal word.

The program will halt after 128 words have been listed. Depress CONT to go on to the next page. Each line in the listing will be preceeded by BELL and a two character address where the line is to be loaded. So, if a tape is cut during the dump, it may later be reloaded by QUICK LOAD.

7700-7753 QL QUICK LOAD -- S-7700

This program may be used to reload any part of memory that was dumped by QUICK DUMP or DRC. The tape dumped by DRC will not contain a starting address; so, you must supply it. Before starting the reader, type BELL followed by the 2 character address where you want to start loading. Using the translate table on the next page, you can translate the octal address into its 2 character equivalent.

Each character loaded will be echoed on the typewriter. But, if you want to mute the echo, you type:

BELL ←8;←

before starting the reader. After the tape has been loaded, you can restore the echo by typing;

BELL ←8B:

SYMBOL TABLE

IN THIS TABLE, EACH OCTAL INSTRUCTION IS FOLLOWED BY ITS EQUIVALENT CHARACTER PAIR, AND THEN BY ITS SIX CHARACTER NUMONIC EQUIVALENT.
NOTE: & = AND ; V = OR

7100	Y	CLL	7240	Z@	CLACMA	7450	\H	SNA	7550	]H	SPA&NA	7000	X	NOP
7101	Y!	CLLIAC	7300	[	CLLCLA	7460	\P	SZAVNL	7600	↑	CLA	7001	X!	IAC
7104	Y$	CLLRAL	7401	\!	NOP	7470	\X	SNA&ZL	7604	↑$	LAS	7002	X"	BSW
7106	Y&	CLLRTL	7402	\"	HLT	7500	]	SMA	7621	↑1	CAM	7004	X$	RAL
7110	Y(	CLLRAR	7404	\$	OSR	7501	]!	MQA	7640	↑@	SZACLA	7006	X&	RTL
7112	Y*	CLLRTR	7410	\(	SKP	7510	](	SPA	7650	↑H	SNACLA	7010	X(	RAR
7120	Y0	STL	7420	\0	SNL	7520	]0	SMAVNL	7700	←	SMACLA	7012	X*	RTR
7200	Z	CLA	7421	\1	MQL	7521	]1	SWP	7701	←!	CLAMQA	7020	X0	CML
7201	Z!	CLAIAC	7430	\8	SZL	7530	]8	SPA&ZL	7710	←%	SPACLA	7040	X@	CMA
7204	Z$	GLK	7440	\@	SZA	7540	]@	SMAVZA	7721	←1	CLASWP	7041	XA	CIA

IN THE FOLLOWING TRANSLATE TABLE, AN OCTAL PAIR IS PLACED DIRECTLY OVER ITS EQUIVALENT CHARACTER

& AND DIRIND	+ TAD DIRIND	I ISZ DIRIND	D DCA DIRIND	C JMS DIRIND	J JMP DIRIND		
0	1	2	3	4	5	6	7
01234567	01234567	01234567	01234567	01234567	01234567	01234567	01234567
!"#$%&'()*+,-./	0123456789:;<=>?	@ABCDEFGHIJKLMNO	PQRSTUVWXYZ[\]↑←				

DECIMAL EQUIVALENT
OF AN OCTAL DIGIT

512	64	8	1
1024	128	16	2
1536	192	24	3
2048	256	32	4
2560	320	40	5
3072	384	48	6
3584	448	56	7

If you want the tape to halt the computer after loading, punch the following characters at the end of the tape:

BELL←+\ "

In the above, we used QUICK LOAD to patch itself. Of course, any program may be patched in the same way. Suppose you want to insert the following patch in your program:

SYMBOLIC LISTING	ODT LISTING	QUICK LOAD CHARACTERS
		BELL
		LI
DCA N2	5451 /3032	8:
TAD I 11	5452 /1411	,)
JMS I PW	5453 /4661	FQ
ISZ N2	5454 /2032	0:

The BELL -- control G -- introduces the address into which the next instruction is to be loaded. The address and the instructions that follow must be translated by you from octal words to character pairs using the translate table on the preceeding page.

```
0200 /6032  tsm,  kcc          / test message
0201 /6040        tfl
0202 /1206        tad  ma
0203 /4607        jms i ames
0204 /7402        hlt
0205 /5202        jmp  .-3
0206 /0210  ma,
0207 /5000  ames,
0210 /7776  cr.
0211 /5657  no
0212 /6700  w
0213 /5163  is
0214 /0064   t
0215 /5045  he
0216 /0064   t
0217 /5155  im
0220 /4500  e
0221 /4657  fo
0222 /6200  r
0223 /4154  al
0224 /5400  l
0225 /4757  go
0226 /5744  od
0227 /0055   m
0230 /4556  en
0231 /7776  cr.
0232 /6457  to
0233 /0043   c
0234 /5755  om
0235 /4500  e
0236 /6457  to
0237 /0064   t
0240 /5045  he
0241 /0041   a
0242 /5144  id
0243 /0057   o
0244 /4600  f
0245 /6450  th
0246 /4551  ei
0247 /6200  r
0250 /6041  pa
0251 /6264  rt
0252 /7100  y
0253 /7777  eof.
```

DUMP REGRESSION COEFFICIENTS

```
4400 /0000    pw,     0               / print word
4401 /3047            dca  ch
4402 /1047            tad  ch
4403 /7002            bsw
4404 /4210            jms  tc
4405 /1047            tad  ch
4406 /4210            jms  tc
4407 /5600            jmp i pw
4410 /0000    tc,     0               / type character
4411 /0250            and  p77
4412 /1215            tad  p240
4413 /4475            type
4414 /5610            jmp i tc
4415 /0240    p240,
4416 /7740    m40,
4417 /7402            hlt
4420 /6032    drc,    kcc             / dump regression coefficients
4421 /6040            tfl
4422 /4474            cr
4423 /1216            tad  m40        / count words / line
4424 /3043            dca  int        / set at  32
4425 /7604            las             / code from switch reg.
4426 /3104            dca  104        / set in 104 for identification
4427 /7327            7327            / 6
4430 /1105            tad  105        / number of coefficients
4431 /7041            cia
4432 /3046            dca  cnt        / set count
4433 /1250            tad  p77        / starting address
4434 /3010            dca  10
4435 /1410            tad i 10
4436 /4200            jms  pw
4437 /2043            isz  int        / 32  words per line
4440 /5244            jmp  .+4
4441 /4474            cr              / carriage return
4442 /1216            tad  m40        / reset words per line
4443 /3043            dca  int
4444 /2046            isz  cnt
4445 /5235            jmp  .-10       / repeat
4446 /7402            hlt             / halt when done
4447 /5246            jmp  .-1
4450 /0077    p77,
```

```
4451 /4474   exit,  cr            / carriage return
4452 /4476          fpnt
4453 /0356          fget  tot     / get total error sq.
4454 /4267          fdiv  num     / calculate average
4455 /0000          fext
4456 /4665          jms i alog    / calculate square root
4457 /7240          cla cma
4460 /1061          tad  exp
4461 /3061          dca  exp
4462 /4666          jms i aantl
4463 /4752          jms i afout   / print root mean square
4464 /5217          jmp  drc-1    / end of run
4465 /6661   alog,
4466 /5271   aantl
4467 /2044   num,
4470 /0000
4471 /0000
4472 /7307   prt2,  7307          / -4    part two
4473 /1104          tad  n4
4474 /3100          dca  n0       / set log test parameter
4475 /3104          dca  n4       / zero n4
4476 /6032          kcc
4477 /6040          tfl
4500 /4476          fpnt
4501 /0754          fget i azro   / initialize
4502 /7356          fput  tot     / total
4503 /7773          fput i anum   / and number
4504 /0000          fext
4505 /4747          jms i awxyz   / read data w, x, y, & z
4506 /5251          jmp  exit
4507 /1353          tad  p77      / set index register to
4510 /4750          jms i aply    / read pars. in  ply
4511 /7567          adx           / address of  x used by  ply
4512 /4474          cr            / issue carriage return
4513 /4476          fpnt
4514 /7771          fput i awc    / save calculated value of  w
4515 /0755          fget i aone
4516 /1773          fadd i anum
4517 /7773          fput i anum   / increment  number
4520 /0771          fget i awc
4521 /2774          fsub i adw
4522 /7772          fput i adif   / calculate error
4523 /3772          fmul i adif   / square it and
4524 /1356          fadd  tot     / add to total
4525 /7356          fput  tot
4526 /0000          fext
4527 /1361          tad  m4       / prepare to
4530 /3046          dca  cnt      / type
4531 /1351          tad  am7      / 7  variables
4532 /3010          dca  10
4533 /1370          tad  m7
4534 /3047          dca  ch       / set counter
4535 /1410          tad i 10      / get next address
4536 /3034          dca  tm       / store it
4537 /4476          fpnt
4540 /0434          fget i tm     / get next variable
4541 /0000          fext
4542 /4746          jms i atypef  / type it
4543 /2047          isz  ch       / check count
4544 /5335          jmp  .-7
4545 /5305          jmp  prt2+13  / return to get next data.
```

```
4546 /7166    atypef,
4547 /7300    awxyz,
4550 /4600    aply,
4551 /4570    am7,
4552 /6200    afout,
4553 /0077    p77,
4554 /4753    azro,
4555 /7550    aone,
4556 /1736    tot,
4557 /1403
4560 /4265
4561 /7774    m4,
4562 /1755    dif,
4563 /0276
4564 /2400
4565 /2056    wc,
4566 /0120
4567 /5745
4570 /7771    m7,
4571 /4565    awc,
4572 /4562    adif,
4573 /4467    anum,
4574 /7564    adw,
4575 /7567    adx,
4576 /7572    ady,
4577 /7575    adz,
4600 /4512    ply,    0              / polynomial expansion routine
4601 /3044            dca  cy        / to calculate a value for  w
4602 /1044            tad  cy        / that matches given values of
4603 /3010            dca  10        / x, y, & z. Set index register
4604 /1344            tad  here      / get address where parameters
4605 /3011            dca  11        / are to be stored in this routine.
4606 /1337            tad  m6        / set counter
4607 /3043            dca  cx
4610 /1410            tad i 10       / move parameters into this routine.
4611 /3411            dca i 11
4612 /2043            isz  cx        / check counter
4613 /5210            jmp  .-3
4614 /7307            7307           / set  cn  at address of the  last
4615 /1044            tad  cy        / regression coefficient.
4616 /1352            tad  n5
4617 /3341            dca  cn
4620 /1336            tad  ax        / set aout = to address where
4621 /3340            dca  aout      / x is to be stored in this routine.
4622 /7346            7346           / -3
4623 /3351            dca  n4        / set counter
4624 /1600            tad i ply      / get address of  x  from
4625 /3356    bck,    dca  fx        / calling routine and store in  fx.
4626 /4476            fpnt
4627 /0756            fget i fx      / get next variable from calling
4630 /0000            fext           / program.
4631 /1351            tad  n4        / check to see if we are
4632 /1345            tad  n0        / to take the logarithm
4633 /7710            spa cla        / skip if no
4634 /4743            jms i alog     / take the logarithm
4635 /4476            fpnt
4636 /7740            fput i aout    / store the variable in this
4637 /0000            fext           / routine.
```

```
4640 /7325          7325          / increment aout by 3
4641 /1340          tad  aout
4642 /3340          dca  aout
4643 /7325          7325          / increment  fx
4644 /1356          tad  fx
4645 /2351          isz  n4       / check count
4646 /5225          jmp  bck      / go back for next variable
4647 /4476          fpnt,
4650 /0353          fget zro
4651 /7364          fput  fz
4652 /0000          fext
4653 /1350          tad  n3
4654 /3045          dca  cz
4655 /1347   lz,     tad  n2       / expand the polynomial
4656 /3044          dca  cy       / equation.
4657 /4476          fpnt
4660 /0375          fget  z
4661 /3364          fmul  fz
4662 /7364          fput  fz
4663 /0353          fget  zro
4664 /7361          fput  fy
4665 /0000          fext
4666 /1346   ly,     tad  n1
4667 /3043          dca  cx
4670 /4476          fpnt
4671 /0372          fget  y
4672 /3361          fmul  fy
4673 /7361          fput  fy
4674 /0353          fget  zro
4675 /7356          fput  fx
4676 /6600          fskp
4677 /4476   lx,     fpnt
4700 /0367          fget  x
4701 /3356          fmul  fx
4702 /1741          fadd i cn
4703 /7356          fput  fx
4704 /0000          fext
4705 /7346          7346
4706 /1341          tad  cn
4707 /3341          dca  cn
4710 /2043          isz  cx
4711 /5277          jmp  lx
4712 /4476          fpnt
4713 /0361          fget  fy
4714 /1356          fadd  fx
4715 /7361          fput  fy
4716 /0000          fext
4717 /2044          isz  cy
4720 /5266          jmp  ly
4721 /4476          fpnt
4722 /0364          fget  fz
4723 /1361          fadd  fy
4724 /7364          fput  fz
4725 /0000          fext
4726 /2045          isz  cz
4727 /5255          jmp  lz
```

```
4730 /7346          7346          / check to see if we are to
4731 /1345          tad  n0       / take the antilogarithm of
4732 /7750          spa sna cla   / the calculated value of  w.
4733 /4742          jms i aantl   / yes
4734 /2200          isz  ply      / increment return address
4735 /5600          jmp i ply     / return.
4736 /4767   ax,
4737 /7772   m6,
4740 /5000   aout,
4741 /0103   cn,
4742 /5271   aantl,
4743 /6661   alog,
4744 /4744   here,
4745 /0001   n0,
4746 /7775   n1,
4747 /7776   n2,
4750 /7777   n3,
4751 /0000   n4,
4752 /0022   n5,
4753 /0000   zro,
4754 /0000
4755 /0000
4756 /2024   fx,
4757 /7637
4760 /0743
4761 /2026   fy,
4762 /2715
4763 /3722
4764 /2026   fz,
4765 /2715
4766 /3722
4767 /2034   x,
4770 /3034
4771 /6177
4772 /2005   y,
4773 /4271
4774 /0311
4775 /2014   z,
4776 /0000
4777 /0000

0043/1744   cx,
0044 /0000   cy,
0045 /0000   cz,
```

```
5000 /0204   mes,    0              / message subroutine
5001 /1217           tad  m1        / subtract 1 from address
5002 /3017           dca  17        / set index register
5003 /1417           tad i 17       / get next word
5004 /3306           dca  chr       / save word
5005 /1306           tad  chr
5006 /2306           isz  chr       / is it  eof ?
5007 /7410           skp
5010 /5600           jmp i mes      / yes. return.
5011 /2306           isz  chr       / no. is it a carriage return ?
5012 /7410           skp
5013 /5220           jmp  .+5       / yes.
5014 /4616           jms i apw      / no. print word.
5015 /5203           jmp  mes+3     / go back for next word.
5016 /4400   apw,                   / address of print word.
5017 /7777   m1,
5020 /4266           jms  cr        / return carriage.
5021 /5203           jmp  mes+3     / go back for next word.

5043/6200    afout,                 / address of output routine.
5044 /4620   s3,
5045 /4206   s2,
5046 /0004   s1,
5047 /2427   s0,
5050 /4453   low,                   / low end of trace range.
5051 /4524   high,                  / high end of trace range
5052 /5405   ret,                   / return address for trace.
5053 /0215   p215,
5054 /5400   afpnt,                 / address of  fpnt
5055 /6305   printc,                / address of print character.
5056 /0212   p212,
5057 /0240   p240,
5060 /5073   care,   0              / basic carriage return
5061 /1253          tad  p215
5062 /4655          jsm i printc
5063 /1256          tad  p212
5064 /4655          jms i printc
5065 /5660          jmp i care
5066 /4423   cr,     0              / carriage return
5067 /7001          iac             / if  -1 , turn page.
5070 /7650          sna cla
5071 /5275          jmp  .+4
5072 /4260          jms  care
5073 /2377          isz  lc        / check line count.
5074 /5666          jmp i cr       / return if not empty
5075 /1377          tad  lc
5076 /1375          tad  m6        / count  6  lines at
5077 /3377          dca  lc        / end of page
5100 /4260          jms  care
5101 /2377          isz  lc        / count to start of next
5102 /5300          jmp  .-2       / page.
5103 /1376          tad  m74       / reset line count at  60.
5104 /3377          dca  lc
5105 /5666          jmp i cr       / return
5106 /0011   chr,
5107 /7774   m4,
```

UTILITY PROGRAMS

```
5110 /5152   poa,    0              / print octal accumulator
5111 /7004           ral
5112 /3306           dca  chr
5113 /1307           tad  m4        / set count for four
5114 /3046           dca  cnt       / characters.
5115 /1306           tad  chr
5116 /7004           ral          / shift next character
5117 /7006           rtl            / into position
5120 /3306           dca  chr
5121 /1306           tad  chr       / save character
5122 /0332           and  p7        / mask digit
5123 /1333           tad  p260      / make it printable
5124 /4655           jms i printc   / type this digit
5125 /2046           isz  cnt       / check count
5126 /5315           jmp  .-11
5127 /1257           tad  p240      / type space
5130 /4655           jms i printc   / when done
5131 /5710           jmp i poa      / and return
5132 /0007   p7,
5133 /0260   p260,
5134 /1654   trace,  tad i afpnt    / get address
5135 /7041           cia            / make it negative
5136 /3247           dca  s0        / save
5137 /1250           tad  low
5140 /1247           tad  s0
5141 /7740           sma sza cla
5142 /5373           jmp dwn        / return if less than low
5143 /1251           tad  high
5144 /1247           tad  s0        / return if beyond high
5145 /7710           spa  cla
5146 /5373           jmp  dwn
5147 /4266           jms  cr        / carriage return
5150 /1654           tad i afpnt  / get address of instruction
5151 /4310           jms  poa       / print octal address
5152 /1053           tad  ac3
5153 /3244           dca  s3        / sav  fac
5154 /1054           tad  ac2
5155 /3245           dca  s2
5156 /1055           tad  ac1
5157 /3246           dca  s1
5160 /1061           tad  exp
5161 /3247           dca  s0
5162 /4643           jms i fout     / type  fac
5163 /1247           tad  s0        / restore  fac
5164 /3061           dca  exp
5165 /1246           tad  s1
5166 /3055           dca  ac1
5167 /1245           tad  s2
5170 /3054           dca  ac2
5171 /1244           tad  s3
5172 /3053           dca  ac3
5173 /2654   dwn,    isz i afpnt    / increment addr.
5174 /5652           jmp i ret      / return
5175 /7772   m6,
5176 /7704   m74,
5177 /7712   lc,
```

POOR MAN'S EDITOR

This program will copy on the typewriter, any tape read by the keyboard. If you set a character code in the switch register , the computer will halt before the selected character is typed .

```
5200 /6040  pme,    tfl
5201 /6032          kcc
5202 /6031          ksf
5203 /5202          jmp  .-1
5204 /6034          krs
5205 /3047          dca  ch
5206 /7604          las
5207 /7710          spa cla
5210 /5217          jmp  .+7
5211 /7604          las
5212 /0227          and  p777
5213 /7041          cia
5214 /1047          tad  ch
5215 /7640          sza cla
5216 /5222          jmp  .+4
5217 /7402          hlt
5220 /7410          skp
5221 /5200          jmp  pme
5222 /1047          tad  ch
5223 /6041          tsf
5224 /5223          jmp  .-1
5225 /6046          tls
5226 /5201          jmp  pme+1
5227 /0777  p777,
```

If you wish to delete the selected character, depress EXAM , CONT.

If you wish to select every character, set SW0 = 1

For example, if you want to copy a PAL source tape with comments to be added at end of each line:

Put 5200 in address register; then set SW 3-11 = 215. Insert tape in reader, and depress CONT .

The computer will halt at end of each line. Turn reader off and depress EXAM , CONT. Type desired comment followed by a carriage return. Turn reader on. Repeat for each line. You must insert the carriage return, even when no comment is required; unless you depress CONT, but not EXAM.

```
5233 /0203 p203,
5234 /5277 fix,    0
5235 /4664         jms i aar1
5236 /4664         jms i aar1
5237 /4664         jms i aar1
5240 /1061         tad  exp
5241 /7041         cia
5242 /1355         tad p200
5243 /7500         sma
5244 /5251         jmp  .+5
5245 /3046         dca  cnt  / no of left shifts required
5246 /4665         jms i aal1/ to justify decimal point.
5247 /2046         isz  cnt
5250 /5246         jmp  .-2
5251 /7600         cla
5252 /3054         dca  ac2  / zero low order
5253 /3053         dca  ac3  / accumulator
5254 /1233         tad  p203
5255 /3061         dca  exp  / exponent for fixed part.
5256 /1051         tad  sgn
5257 /7104         cll  ral  / put sign in link
5260 /1055         tad  ac1
5261 /7430         szl
5262 /7041         cia       / accumulator now holds fixed
5263 /5634         jmp i fix / part, return.
5264 /5600 aar1,
5265 /6050 aal1,
5266 /0201 p201.
5267 /7777 m1,
5270 /5400 afpnt,
```

antl = ANTI LOG_e = $e^{(X)}$

```
5271 /1664 antl,  0
5272 /4670        jms i afpnt
5273 /3375        fmul  log2e
5274 /7040        fput  f
5275 /0000        fext
5276 /4234        jms  fix
5277 /7510        spa
5300 /1267        tad  m1
5301 /3047        dca  ch       / store characteristic
5302 /4670        jms i afpnt   / fix - f =
5303 /2040        fsub  f       / negative mantissa
5304 /7040        fput  f
5305 /6740        fsgt          / compliment if diff. is
5306 /5312        fjmp .+4      / positive. (mantissa is neg)
5307 /0372        fget  one
5310 /2040        fsub  f
5311 /7040        fput  f
5312 /0000        fext
5313 /3051        dca  sgn      / make mantissa positive.
5314 /1047        tad  ch
5315 /1266        tad  p201
5316 /3061        dca  exp
5317 /1354        tad  p4       / fac  now contains
5320 /3055        dca  ac1      / 2 to  N th  power
5321 /3054        dca  ac2
5322 /3053        dca  ac3
5323 /4670        jms i afpnt
5324 /7043        fput  int
5325 /0040        fget  f
5326 /0000        fext
5327 /3051        dca  sgn
5330 /4670        jms i afpnt   / using formulas on
5331 /3356        fmul  ln202   / page  8-26  .of
5332 /7035        fput  y       / "introduction to prog."
5333 /3035        fmul  y       / we now calculate the
5334 /1361        tad  b1       / r-th  power of  2.
5335 /7040        fput  f
5336 /0364        fget  a
5337 /4040        fdiv  f
5340 /2035        fsub  y
5341 /1367        fadd  a0
5342 /7040        fput  f
5343 /0035        fget  y
5344 /1035        fadd  y
5345 /4040        fdiv  f
5346 /1372        fadd  one
5347 /7040        fput  f
5350 /3040        fmul  f
5351 /3043        fmul  int
5352 /0000        fext
5353 /5671        jmp i antl
5354 /0004 p4,
5355 /0200 p200,
```

```
5356 /1775 ln2o2,          / .34657359
5357 /4271
5360 /0273
5361 /2067 b1,             / 60.090191
5362 /4056
5363 /1323
5364 /6124 a1,             /-601.80427
5365 /5471
5366 /5707
5367 /2046 a0              / 12.015017
5370 /0036
5371 /6017
5372 /2014 one,            / 1.
5373 /0000
5374 /0000
5375 /2015 log2e           / 1.442695
5376 /6125
5377 /0722
```

```
6502 /0262   rint,   0              / read integer
6503 /3061           dca  exp       / initialize variables
6504 /3062           dca  dot
6505 /3051           dca  sgn
6506 /3053           dca  ac3
6507 /3054           dca  ac2
6510 /3055           dca  ac1
6511 /7240           cla  cma
6512 /3046           dca  cnt
6513 /6031   loop,    ksf
6514 /5313           jmp  .-1
6515 /6036           krb
6516 /3047           dca  ch
6517 /1047           tad  ch
6520 /1357           tad  m212      / is it a linefeed ?
6521 /5610           jmp i a7410    / patch to count line feeds
6522 /1047           tad  ch
6523 /1363           tad  m256
6524 /7440           sza            / decimal point ?
6525 /5331           jmp  .+4       / no.
6526 /7240           cla cma        / yes.
6527 /3062           dca  dot       / dot = -1
6530 /5313           jmp  loop
6531 /7001           iac
6532 /7440           sza            / minus sign ?
6533 /5337           jmp  .+4       / no.
6534 /7330           7330           / 4000
6535 /3051           dca  sgn       / set sign = 4000
6536 /5313           jmp  loop
6537 /1365           tad  m3
6540 /7510           spa            / greater than  260 ?
6541 /5313           jmp  loop      / no.  get next digit.
6542 /1364           tad  m12       / yes
6543 /7700           sma  cla       / less than  272 ?
6544 /5702           jmp i rint     / return if no.
6545 /4760           jms i am10     / yes. mult. fac by  10.
6546 /1047           tad  ch        / recover character.
6547 /1362           tad  m260      / make it a digit
6550 /3056           dca  op3       / put in operand.
6551 /3057           dca  op2
6552 /3060           dca  op1
6553 /4761           jms i aoadd    / fac = fac + operand
6554 /1062           tad  dot       / decriment count
6555 /1046           tad  cnt       / if a point (.) was encountered.
6556 /5312           jmp  loop-1    / get next digit.
6557 /7566   m212,                  /  minus  212
6560 /6153   am10,
6561 /5633   aoadd,
6562 /7520   m260,
6563 /7522   m256,
6564 /7766   m12,
6565 /7775   m3,
```

```
6600 /0033 fin,    0              / floating input routine
6601 /4657         jms i arint    /  read integer
6602 /1255         tad  p233      / initialize exponent
6603 /3061         dca  exp
6604 /1055         tad  ac1       / shift fac left 1 to fill
6605 /1254         tad  m4        / low order dig. in ac1
6606 /7700         sma  cla
6607 /5214         jmp  dwn       / when finished
6610 /4650         jms i aal1
6611 /7240         cla  cma
6612 /1061         tad  exp
6613 /5203         jmp  .-10      / loob back
6614 /4342 dwn,    jms  mar       / move fac right to put only
6615 /2046         isz  cnt       / one digit in ac1.
6616 /7410         skp            / count digits right of dec. point
6617 /5600         jmp i fin      / return when done.
6620 /1253         tad  m11
6621 /3261         dca  c11       / c11 = 11
6622 /1053         tad  ac3       / move fac to operand and clear
6623 /3056         dca  op3
6624 /3053         dca  ac3
6625 /1054         tad  ac2
6626 /3057         dca  op2
6627 /3054         dca  ac2
6630 /1055         tad  ac1
6631 /3060         dca  op1
6632 /3055         dca  ac1
6633 /4651 bck,    jms i aar1     / multiply  fac  by  16/10
6634 /4651         jms i aar1
6635 /4651         jms i aar1
6636 /4652         jms i aoadd
6637 /4651         jms i aar1
6640 /4652         jms i aoadd
6641 /2261         isz  c11
6642 /5233         jmp bck        / loop bck 9 times.
6643 /1254         tad  m4        / subtract 4 from exponent
6644 /1061         tad  exp       / each time
6645 /3061         dca  exp
6646 /5214         jmp  dwn
6647 /0000
6650 /6050 aal1,
6651 /5600 aar1,
6652 /5633 aoadd,
6653 /7767 m11,
6654 /7774 m4,
6655 /0233 p233,
6656 /0000
6657 /6502 arint
6660 /7770 m10
```

```
6661 /0000 log,     0           / calculate naperian logarithm
6662 /4752          jms i afpnt
6663 /7040          fput  f     / save argument
6664 /0000          fext
6665 /1061          tad  exp    / get exponent of arg.
6666 /7550          spa sna     / halt if zero or neg.
6667 /7402          hlt
6670 /1353          tad  m200   / change exp to + or - .
6671 /3200          dca  tm
6672 /1354          tad  p203   / initialize exponent
6673 /3061          dca  exp    / of integer part
6674 /7330          g4000
6675 /0200          and  tm
6676 /3051          dca  sgn    / insert proper sign
6677 /3053          dca  ac3
6700 /3054          dca ac2     / put original exponent
6701 /1200          tad  tm     / in fac
6702 /7510          spa
6703 /7041          cla
6704 /3055          dca  ac1
6705 /4342          jms  mar    / justify  fac  right
6706 /4752          jms i afpnt
6707 /7043          fput  int   / store integer part
6710 /0040          fget  f
6711 /0000          fext
6712 /1355          tad  p200
6713 /3061          dca  exp    / mantissa exponent is zero.
6714 /3051          dca  sgn    / sign is also zero
6715 /4752          jms i afpnt
6716 /7040          fput  f     / store mantissa
6717 /1364          fadd  sqh
6720 /7035          fput  z     / z = f + sqrt(.5)
6721 /0040          fget  f     / see "Introduction to Prog."
6722 /2364          fsub  sqh   / page 8-27 .
6723 /4035          fdiv  z     / z = (f - sqrt(.5))/ z
6724 /7035          fput  z
6725 /3035          fmul  z
6726 /7040          fput  f     / f = z**2
6727 /0375          fget c5
6730 /3040          fmul  f
6731 /1372          fadd  c3
6732 /3040          fmul  f
6733 /1367          fadd  c1
6734 /3035          fmul  z     / fac = ((c5*f + c3)*f +
6735 /2356          fsub  phlf  /      c1)*z - .5 + int
6736 /1043          fadd  int
6737 /3361          fmul  ln2   / convert to naperian base
6740 /0000          fext
6741 /5661          jmp i log   / return
```

```
6742 /6615 mar,    0           / / move  fac  right to empty
6743 /1055         tad  ac1    / all bits left of bit 9 in ac1.
6744 /1260         tad  m10
6745 /7710         spa  cla    / are we through ?
6746 /5742         jmp i mar   / yes. return
6747 /4652         jms i aar1  / no. shift fac right.
6750 /2061         isz  exp    / correct ecponent
6751 /5343         jmp  .-6
6752 /5400 afpnt,             / address of f.p.inter.
6753 /7600 m200,
6754 /0203 p203,
6755 /0200 p200,
6756 /1777 phlf,              / .5
6757 /7777
6760 /7774 m4,
6761 /2005 ln2,               / .6931472
6762 /4271
6763 /0276
6764 /2005 sqh,              / .7071068
6765 /5202
6766 /3627
6767 /2025 c1,               / 2.8853913
6770 /6125
6771 /0776
6772 /2007 c3,               / .9614706
6773 /5421
6774 /3571
6775 /2004 c5,               / .59897865
6776 /6253
6777 /2516
```

SOLVE SIMULTANEOUS EQUATIONS

fortran program

```
0073 /5110  poa,                    do 10 k=1,n
0074 /5067  cr,                     f = x(k,k)
0075 /6305  type,                   do 10 j = k,n
0076 /5400  fpnt,                   x(k,j+1) + x(k,j+1)/ f
0077 /5134  link,                   do 10 i = 1,n
0100 /7775  n,                      if( i .eq. k) go to 10
0101 /0000                          x(i,j+1) = x(i,j+1) - x(k,j+1)* x(i,k)
0102 /0000                    10    continue
0103 /0000
0104 /0000
0105 /0011  m,


7000 /0000  ic,
7001 /0000  jc,
7002 /0302  kj,
7003 /0274  ij,
7004 /0000  kc,
7005 /0252  ok,
7006 /0260  kk,
7007 /0263  ik,
7010 /0000
7011 /0236  at,
7012 /5023  sse,   0          / solve simultaneous
7013 /1211         tad at
7014 /3206         dca  kk    / initialize addresses
7015 /1100         tad  n
7016 /3204         dca  kc
7017 /1211         tad  at
7020 /3205  k,     dca  ok
7021 /1205         tad  ok
7022 /1105         tad  m
7023 /3203         dca  ij
7024 /4476         jms i  fpnt
7025 /0606         get i kk   / get term on diagonal
7026 /7040         fput  f
7027 /0000         fext
7030 /1206         tad  kk
7031 /1105         tad  m
7032 /3202         dca  kj
7033 /1204         tad  kc
7034 /3201         dca  jc
7035 /4476  j,     jms i  fpnt
7036 /0602         fget i kj
7037 /4040         fdiv  f    / divide key row by term on diagonal
7040 /7602         fput i kj
7041 /0000         fext
7042 /1205         tad  ok
7043 /3207         dca  ik
7044 /1100         tad  n
7045 /3200         dca  ic
```

```
7046 /1200 ii,    tad  ic       / skip operation if i = k
7047 /7041        cia
7050 /1204        tad  kc
7051 /7650        jmp  cla
7052 /5263        jmp  sk
7053 /4476        fpnt
7054 /0602        fget i kj     / generate zeros on key column
7055 /3607        fmul i ik
7056 /7043        fput  int
7057 /0603        fget i ij
7060 /2043        fsub  int
7061 /7603        fput i ij
7062 /0000        fext
7063 /7325  sk,    7325          / 3
7064 /1203        tad  ij       / increment  ij
7065 /3203        dca  ij
7066 /7325        7325          / 3
7067 /1207        tad  ik       / increment  ik
7070 /3207        dca  ik
7071 /2200        isz  ic       / iterate down column
7072 /5246        jmp  ii
7073 /1105        tad  m
7074 /1202        tad  kj       / increment  kj
7075 /3202        dca  kj
7076 /2201        isz jc        / iterate across row
7077 /5235        jmp  j
7100 /2204        isz  kc
7101 /7410        skp
7102 /5315        jmp  ma
7103 /7325        7325          / 3
7104 /1105        tad  m
7105 /1206        tad  kk       / increment  kk
7106 /3206        dca  kk
7107 /1205        tad  ok
7110 /1105        tad  m        / go back to sweep matrix again
7111 /5220        jmp  k
7112 /7772  m6,
7113 /0000
7114 /0000
```

```
7115 /1105   ma,    tad  m    / move answers to addr  106
7116 /7041          cia
7117 /3204          dca  kc
7120 /7305          7305 kk   / 2
7121 /1206          tad  kk
7122 /3010          dca  10
7123 /1364          tad  p105
7124 /3011          dca  11
7125 /1410          tad i 10
7126 /3411          dca i 11
7127 /2204          isz  kc
7130 /5325          jmp  .-3
7131 /7600          cla
7132 /1312          tad  m6
7133 /3204          dca  kc
7134 /4474          jms i cr
7135 /1363          tad  p77
7136 /3010          dca  10
7137 /1410          tad i 10
7140 /4473          jms  poa  / print six parameters in octal
7141 /2204          isz  kc
7142 /5337          jmp  .-3
7143 /7240          cla  cma
7144 /3046          dca  cnt
7145 /1100          tad  n
7146 /3204          dca  kc
7147 /2010          isz  10
7150 /4476          jms i fpnt
7151 /0410          fget i 10 / print answers
7152 /0000          fext
7153 /2010          isz  10
7154 /2010          isz  10
7155 /4366          jms i typef
7156 /2204          isz  kc
7157 /5347          jmp  .-10
7160 /5612          jmp i sse
7161 /6200   afout,
7162 /7774   m4,
7163 /0077   p77,
7164 /0105   p105,
7165 /0240   p240,
7166 /7156   typef,0          / type four numbers per line
7167 /2046          isz  cnt
7170 /5374          jmp  .+4
7171 /4474          jms i cr
7172 /1362          tad  m4
7173 /3046          dca  cnt
7174 /4761          jms  afout
7175 /1365          tad  p240
7176 /4475          jms i type
7177 /5766          jmp i typef
```

TRACE OF SSE Solve simultaneous equations

START 5020G

```
7061  1.000000E+00
7025  1.500000E+00
7026  1.500000E+00
7036  1.000000E+00
7037  6.666667E-01
7040  6.666667E-01
7054  6.666667E-01
7055  1.000000E+00
7056  1.000000E+00
7057  2.000000E+00
7060  1.000000E+00
7061  1.000000E+00
7054  6.666667E-01
7055  6.666667E-01
7056  6.666667E-01
7057  1.000000E+00
7060  3.333333E-01
7061  3.333333E-01
7036  5.000000E-01
7037  3.333333E-01
7040  3.333333E-01
7054  3.333333E-01
7055  5.000000E-01
7056  5.000000E-01
7057  5.000000E-01
7060  0
7061  0
7054  3.333333E-01
7055  3.333333E-01
7056  3.333333E-01
7057  1.000000E+00
7060  6.666667E-01
7061  6.666667E-01
7025  3.333333E-01
7026  3.333333E-01
7036  6.666667E-01
7037  2.000000E+00
7040  2.000000E+00
7054  2.000000E+00
7055  2.000000E+00
7056  2.000000E+00
7057  0
7060 -2.000000E+00
7061 -2.000000E+00
7054  2.000000E+00
7055  1.333333E+00
7056  1.333333E+00
7057  3.333333E-01
7060 -1.000000E+00
7061 -1.000000E+00
7775 0000 0000 0000 0000 0011
7151 -2.000000E+00-2.000000E+00
7151 -1.000000E+00-1.000000E+00
7151  2.000000E+00 2.000000E+00
7025  2.000000E+00
7026  2.000000E+00
7036  3.000000E+00
7037  1.500000E+00
7040  1.500000E+00
7054  1.500000E+00
7055  4.500000E+00
7056  4.500000E+00
7057  6.000000E+00
7060  1.500000E+00
7061  1.500000E+00
7054  1.500000E+00
7055  6.000000E+00
7056  6.000000E+00
7057  7.000000E+00
7060  1.000000E+00
7061  1.000000E+00
7036  4.000000E+00
7037  2.000000E+00
7040  2.000000E+00
7054  2.000000E+00
7055  6.000000E+00
7056  6.000000E+00
7057  7.000000E+00
7060  1.000000E+00
7061  1.000000E+00
7054  2.000000E+00
7055  8.000000E+00
7056  8.000000E+00
7057  9.000000E+00
7060  1.000000E+00
7061  1.000000E+00
7036  1.000000E+00
7037  5.000000E-01
7040  5.000000E-01
7054  5.000000E-01
7055  1.500000E+00
7056  1.500000E+00
7057  2.000000E+00
7060  5.000000E-01
7061  5.000000E-01
7054  5.000000E-01
7055  2.000000E+00
7056  2.000000E+00
7057  3.000000E+00
7060  1.000000E+00
```

TEST MATRIX

-2	-1	2	
2	3	4	1
3	6	7	2
4	7	9	3

MULTIPLE REGRESSION PROGRAM

```
7200 /6032  ra1,   kcc            / clear & set flags
7201 /6040         tfl
7202 /4666         jms i arint    / get number
7203 /1047         tad  ch        / test termination
7204 /1373         tad  m305      / is it e?
7205 /7450         sna
7206 /5222         jmp  .+14      / yes. end of parameters
7207 /7500         sma
7210 /5202         jmp  ra1+2     / go back if greater than  e.
7211 /1372         tad  p5        / is it less than  a.
7212 /7550         spa sna
7213 /5202         jmp  ra1+2     / yes. go back for next char.
7214 /1371         tad  p3100     / no. fix up a store instruction.
7215 /3220         dca  .+3
7216 /1053         tad  ac3       / get the integer
7217 /7041         cia            / make it negative
7220 /3102         3101           / store it.
7221 /5202         jmp  ra1+2     / get next parameter.
7222 /1101         tad  n1
7223 /7500         sma            / get first parameter.
7224 /7402         hlt            / halt if positive
7225 /7041         cia            / make it positive
7226 /3105         dca  n5        / load in multiplicand reg.
7227 /1102         tad  n2        / get second parameter.
7230 /4267         jms mli        / multiply  n1*n2
7231 /1103         tad  n3        / get third parameter
7232 /4267         jms  mli       / multiply it also
7233 /1105         tad  n5
7234 /7041         cia
7235 /3100         dca  n0        / -(n1*n2*n3)
7236 /7346         7346           / -3
7237 /4267         jms  mli       / multiply prod by  3
7240 /1100         tad  n0        / no of degrees of freedom
7241 /1370         tad  p34
7242 /7710         spa cla
7243 /7402         hlt            / halt if prod. .gt. 28
7244 /1100         tad  n0
7245 /3046         dca  cnt       / set count = n0
7246 /1105         tad  n5        / multiply by words per column
7247 /2046         isz  cnt
7250 /5246         jmp  .-2
7251 /1105         tad  n5
7252 /7041         cia
7253 /3046         dca  cnt       / count = no. of words in
7254 /1367         tad  p235      / matrix. set index register
7255 /3010         dca  10
7256 /3410         dca i 10       / clear entire matrix
7257 /2046         isz  cnt
7260 /5256         jmp  .-2
7261 /4766         jms i areg     / form regression matrix
7262 /7300         cll  cla
7263 /4765         jms i asse     / solve simultaneous equations
7264 /7402         hlt            / halt when done
7265 /5764         jmpi aprt2     / jump to part 2
```

```
7266/6502    arint,              / address of read integer routine
7267 /7240   mli,    0           / multiply integer routine
7270 /7500           sma
7271 /7402           hlt         / halt if count pos.
7272 /3046           dca  cnt
7273 /1105           tad  n5
7274 /2046           isz  cnt
7275 /5273           jmp  .-2
7276 /3105           dca  n5     / put product in parameter  5
7277 /5667           jmp i mli   / return
7300 /4506   wxyz,   0           / read and store w, x, y, or z
7301 /4761           jms i afin  / get floating point numb. from
7302 /1047           tad  ch     / keyboard. test terminating charac.
7303 /1363           tad  m333
7304 /7450           sna
7305 /5326           jmp i wxyz  / end line if = 333
7306 /7500           sma
7307 /5700           jmp i wxyz  / end of file if greater than 333
7310 /1372           tad  p5     / if = or less than 326, go back
7311 /7550           spa  sna    / for next character.
7312 /5301           jmp  wxyz+1 / fix up a floating point point
7313 /1362           tad  p7773 /inst. to store numb. in address
7314 /3323           dca  ins    / specified by term. character
7315 /7307           7307        / 4
7316 /1104           tad  n4     / check  n4  to determine if we
7317 /1323           tad  ins    / are to take log.
7320 /7510           spa
7321 /4760           jms i log
7322 /4476           fpnt
7323 /7775   ins,    7775        / store number
7324 /0000           fext
7325 /5301           jmp  wxyz+1 / go back for next number.
7326 /7000           nop         / change to a hlt if you want to edit
7327 /7410           skp         / the input. EXAM, CONT  will
7330 /5301           jmp  wxyz+1 / overwrite the current line.
7331 /2300           isz  wxyz   / not end of file; so skip.
7332 /5700           jmp i wxyz  / and return.
7333 /0000
7334 /0000
7335 /0000
7336 /0000
7337 /0000
7340 /5177   alc,                / address of line count
7341 /7550   aone,              / address of  one
7342 /4476   ra,     fpnt        / regress analysis  program.
7343 /0741           fget i aone /
7344 /7775           fput i ax   / set parameters at default
7345 /7776           fput i ay   / values.
7346 /7777           fput i az
7347 /0000           fext
7350 /7240           cla cma
7351 /3740           dca i alc   / set linecount at  -1  to break
7352 /7240           cla  cma    / page at start of problem.
7353 /3102           dca  n2
7354 /7240           cla  cma
7355 /3103           dca  n3
7356 /3104           dca  n4
7357 /5200           jmp  ra1
```

```
7360 /6661    alog,             / address of log routine
7361 /6600    afin,             / address of floating input rout.
7362 /7773    p7773,            / plus  7773
7363 /7445    m333,             / minus  333
7364 /4472    aprt2,            / address of part two
7365 /7012    asse,             / address of solve simul. equations.
7366 /7413    areg,             / address to build regression matrix.
7367 /0235    p235,             / plus  235
7370 /0034    p34,              / plus   34
7371 /3100    p3100,
7372 /0005    p5,
7373 /7473    m305,
7374 /7564    aw,               / address of w
7375 /7567    ax,               / address of x.
7376 /7572    ay,               / address of y
7377 /7575    az,               / address of z
7400 /0000
7401 /0000
7402 /6522    a6522,
7403 /4474            cr
7404 /5602            jmp i a6522
7405 /1047            tad  ch
7406 /4475            type      / this patch was
7407 /5602            jmp i a6522
7410 /7640            sza  cla  / called from  6521  to
7411 /5205            jmp  .-4  / count line feeds entered
7412 /5203            jmp  .-6  / with the input.
7413 /7262    reg,    0         / routine to build regression matrix
7414 /4476            fpnt      / enter floating point interpreter
7415 /0350            fget  one
7416 /7361            fput  vc
7417 /0000            fext      / exit floating point interpreter.
7420 /1341            tad  p106
7421 /3005            dca  uk   / address of subscripted var. u(k).
7422 /4743            jms i awxyz  / get w, x, y, z, from keyboard.
7423 /5613            jmp i reg    / return if end of file
7424 /1103            tad  n3      / set up first row of matrix in u(k)
7425 /3347            dca  cc      / set c count.
```

```
7426 /1102   lc,    tad  n2        / loop  c
7427 /3346          dca  bc        / set  b count
7430 /4476          fpnt
7431 /0350          fget  one
7432 /7356          fput  vb
7433 /6600          fskp
7434 /4476   lb,    fpnt           / loop  b
7435 /0350          fget  one
7436 /7353          fput  va
7437 /0000          fext
7440 /1101          tad  n1
7441 /3345          dca  ac        / set a count
7442 /4476          fpnt
7443 /0353          fget  va
7444 /3356          fmul  vb
7445 /3361          fmul  vc
7446 /7405          fput i uk      / place value in  u(k)
7447 /0353          fget  va
7450 /3367          fmul  x
7451 /7353          fput  va
7452 /0000          fext
7453 /7325          7325           / 3
7454 /1005          tad  uk        / increment to next tripple precision
7455 /3005          dca  uk        / word.
7456 /2345          isz  ac        / check a count
7457 /5242          jmp  la        / repeat loop  a
7460 /4476          fpnt
7461 /0356          fget  vb
7462 /3372          fmul  y
7463 /7356          fput  vb
7464 /0000          fext
7465 /2346          isz  bc        / check  b  count
7466 /5234          jmp  lb        / repeat loop b
7467 /4476          fpnt
7470 /0361          fget  vc
7471 /3375          fmul  z
7472 /7361          fput  vc
7473 /0000          fext
7474 /2347          isz  cc        / check  c  count
7475 /5226          jmp  lc        / repeat loop  c
7476 /4476          fpnt
7477 /0364          fget  w
7500 /7405          fput i uk      / put  w  in last term of  u(k)
7501 /0000          fext
7502 /1342          tad  p236      / set first address of matrix
7503 /3006          dca  xl        / in  xl
7504 /7040          cma            / subtract  1  from negative count
7505 /1100          tad  n0        / in  n0  to process (n+1) terms.
7506 /3346          dca  mc
7507 /1341          tad  p106      / set up first address of variable  u.
```

```
7510 /3003   ol,     dca  um       / store next address of  u(m)
7511 /1100           tad  n0       / in  um.  to begin outer loop.
7512 /3345           dca  lc       / set  l  count
7513 /1341           tad  p106     / initialize address
7514 /3007           dca  ul       / ul
7515 /4476   il,     fpnt          / inner loop
7516 /0407           fget i ul     / get u(l)
7517 /3403           fmul i um     / multiply by  u(m)
7520 /1406           fadd i xl     / add to  x(l)
7521 /7406           fput i xl     / and update  x(l)
7522 /0000           fext
7523 /7325           7325          / 3
7524 /1007           tad  ul       / increment address  ul
7525 /3007           dca  ul
7526 /7325           7325          / 3
7527 /1006           tad  xl       / increment address  xl
7530 /3006           dca  xl
7531 /2345           isz  lc       / check  l  count
7532 /5315           jmp  il       / repeat inner loop
7533 /7325           7325          / 3
7534 /1003           tad  um       / increment address  um
7535 /2346           isz  mc       / check  m  count
7536 /5310           jmp  ol       / repeat outer loop
7537 /5214           jmp  reg+1    / go back to get next set of variables.
7540 /0000
7541 /0106   p106,
7542 /0236   p236,
7543 /7300   awxyz,                / address of routine   wxyz
7544 /0000
7545 /0000   ac,                   / a count.  lc=ac
7546 /0000   bc,                   / b count.  mc = bc
7547 /0000   cc,                   / c count
7550 /2014   one,                  / floating constant -- unity
7551 /0000
7552 /0000
7553 /2077   va,                   / variable  a
7554 /6400
7555 /0000
7556 /2014   vb,                   / variable  b
7557 /0000
7560 /0000
7561 /2014   vc,                   / variable  c
7562 /0000
7563 /0000
7564 /2056   w,                    / variable  w
7565 /4000
7566 /0000
7567 /2035   x,                    / variable  x
7570 /0000
7571 /0000
7572 /2014   y,                    / variable  y
7573 /0000
7574 /0000
7575 /2024   z,                    / variable  z
7576 /0000
7577 /0000

                  um=03
                  uk=05
                  xl=06
                  ul=07
```

QUICK DUMP

```
7600 /6032 qd,,   kcc          / quick dump
7601 /6040        tfl
7602 /7604        las          / get first address to be dumped.
7603 /1267        tad  m1      / subtract  1
7604 /3010        dca  10      / put in index register
7605 /1270        tad  m4
7606 /3272        dca  c4      / c4 + -4
7607 /4255        jms  cr      / return carriage
7610 /4255        jms  cr
7611 /4255        jms  cr      / nop these two instructions
7612 /4255        jms  cr      / to single space the output.
7613 /1275        tad  m40     / set counter at
7614 /3271        dca  c1      / 32 words per line
7615 /1265        tad  p207    / start each line with a bell.
7616 /4232        jms  tp
7617 /1010        tad  10      / and address of first word.
7620 /7001        iac
7621 /4240        jms  pw      / print address of first word
7622 /1410        tad i 10     / get next word
7623 /4240        jms  pw      / and type or print word
7624 /2271        isz  c1      / end of line ?
7625 /5222        jmp  .-3
7626 /2272        isz  c4      / yes. quit after four lines
7627 /5210        jmp  qd+10
7630 /7402        hlt          / one page has been dumped
7631 /5205        jmp  qd+5    / return to print next page
7632 /7654  tp,   0            / type character in accumulator
7633 /6046        tsf
7634 /6041        jmp  .-1
7635 /5234        tls
7636 /3377        dca  chr     / save character
7637 /5632        jmp i tp     / return
7640 /7624  pw,   0            / print word
7641 /3346        dca  tm      / save word
7642 /1346        tad  tm
7643 /7002        bsw
7644 /4250        jms  tc      / type first character
7645 /1346        tad  tm
7646 /4250        jms  tc      / type second character
7647 /5640        jmp i pw     / return
7650 /7645  tc,   0            / type character
7651 /0266        and  p77     / retain low order chara
7652 /1274        tad  p240    / format for printing
7653 /4232        jms  tp      / type it
7654 /5650        jmp i tc     / return
```

```
7655 /7613  cr,    0              / carriage return
7656 /7200         cla
7657 /1273         tad  p215
7660 /4232         jms  tp
7661 /1264         tad  p212
7662 /4232         jms  tp
7663 /5655         jmp i cr
7664 /0212  p212,
7665 /0207  p207,
7666 /0077  p77,
7667 /7777  m1,
7670 /7774  m4,
7671 /7756  c1,
7672 /7774  c4,
7673 /0215  p215,
7674 /0240  p240,
7675 /7740  m40,
7676 /0100  p100,
7677 /7700  m100,
7700 /6032  ql,    kcc            / quick load
7701 /6040         tfl
7702 /1353         tad  i5310
7703 /3313         dca  gw-2      / restore last instruction
7704 /4315         jms  gw        / get next word from keyboard
7705 /5304         jmp  .-1       / return for address if a bell was heard.
7706 /1267         tad  m1        / subtract  1  from address
7707 /3010         dca  10        / put in index register
7710 /4315         jms  gw        / get next word.
7711 /5304         jmp  .-5       / return for address if bell was heard.
7712 /3410         dca i 10       / load word in core.
7713 /5310         jmp  .-3       / go back for next word
7714 /7571  m207,
7715 /7711  gw,    0
7716 /7240         cla cma
7717 /4326         jms  rd        / read first character
7720 /7002         bsw            / switch to high order position
7721 /7421         mql            / save it
7722 /4326         jms  rd        / get next character
7723 /7501         mqa            / combine two characters
7724 /2315         isz  gw        / since bell was not heard, skip.
7725 /5715         jmp i gw       / return
```

```
7726 /7723   rd,    0              / read characters
7727 /4346          jms  re
7730 /4232          jms  tp
7731 /1377          tad  chr
7732 /1314          tad  m207      / is it a bell ?
7733 /7450          sna            / yes--(return without a skip)
7734 /5715          jmp i gw
7735 /1345          tad  m31       / no -- check is it less than 240.
7736 /7510          spa
7737 /5327          jmp  rd+1      / yes. ignore and get next char.
7740 /1277          tad  m100
7741 /7500          sma            / is it greater than 337
7742 /5327          jmp  rd+1      / yes, ignore.
7743 /1276          tad  p100      / restore character
7744 /5726          jmp i rd       / return
7745 /7747   m31,
7746 /7730   re,    0              / get next character
7747 /6031          ksf
7750 /5347          jmp  .-1
7751 /6036          krb
7752 /5746          jmp i re
7753 /5310   i5310,
7754 /0204
7755 /0022                         / the first instruction in the
7756 /5300                         / rim loader will be overwritten
                                   / with a transfer to ql  whereupon
                                   / the instruction will be restored
                                   / and than a halt will be generated
                                   / by  bell ←NP:
                                   /     bell ←+\"  at end of  tape.
```

A P P E N D I X

EXAMPLES

We will now present two examples to show how the program works.

The first, called SMALL TEST , is a very short problem involving a logarithmic fit.

The second, called BIG TEST , is a fair sized problem (4A, 4B, or 16 degrees of freedom). In this problem, we are asking for a formula with which the computer may calculate the enthalpy of steam when pressure, and temperature are given.The data was taken from, "Thermodynamic Properties of Steam" , by Keenan and Keys . John Wiley & Sons -- 1936.

In this example, we specified 16 degrees of freedom, But, we ran the problem 4 ways to show how the Standard Deviation varies when we specify more degrees of freedom. Results are given in the following table:

A	B	Standard Deviation
4	4	.000423
4	5	.000238
9	3	.000128
7	4	.000115

The computer took less than 40 seconds to solve 28 simultaneous equations.It expanded the 28 degree polynomial equation in less time than it takes for a carriage return.

From statistics, we learn that , with a normal distribution and an infinite number of points, 68.26% of all cases will be within one standard deviation from the mean, and 95.46% will be within two standard deviations. In our problem, with 45 points, no point exceeded two deviations (.00085) ; but, one point -- number 18 -- was very close. In any case, the accuracy of the formula is about what you could get by a manual reading of the Steam Chart.

We define symbols as follows:

SYMBOL	PROPERTY	UNITS	RANGE
W	ENTHALPY	Thous Btu/lb	1.4 - 1.52
X	ABS. PRES.	Thous psia.	.54- .62
Z	TEMPERATURE	Hund. Deg. F	8.0 -10.0

To avoid exponent overflow, it was necessary to scale units near unity. For example, in the problem having 28 degs. of freedom, the pressure is raised to the twelth power and multiplied by temperature to the sixth power. This product exceeded the range or our exponent(10^{38}) when we tried to correlate to a natural scale.

REGRESSION ANALYSIS EXAMPLE -- SMALL TEST

Start at address 7342

DATA

```
3A   2D   E
2.5W   1.X   [
2.9W   2.X   [
5.W   4.X   [
12.W  8.X   [\      PARAMETERS AND REGRESSION COEFFICIENTS
7775 7775 7777 7777 7776 0011
 9.130124E-01 -2.918390E-02  3.783139E-01
```

Start after halt at address 7265

RESULTS

V W	(V - W) X	N Y	Z

```
3A   2D   E
2.5W   1.X   [
 2.491817E+00 -8.182645E-03  1.000000E+00
 2.500000E+00  1.000000E+00  1.000000E+00  1.000000E+00
2.9W   2.X   [
 2.928662E+00  2.866203E-02  2.000000E+00
 2.900000E+00  2.000000E+00  1.000000E+00  1.000000E+00
5.W   4.X   [
 4.951067E+00 -4.893285E-02  3.000000E+00
 5.000000E+00  4.000000E+00  1.000000E+00  1.000000E+00
12.W  8.X   [
 1.203940E+01  3.940141E-02  4.000000E+00
 1.200000E+01  8.000000E+00  1.000000E+00  1.000000E+00 \
 3.476840E-02
```

RMS or Standard Devistion

Start after halt at address 4420

DRC = Dump regression Coefficients. QUICK LOAD will put these coefficients in your program.

ST = Small Test. Put in switch register before start at address 4420. See translate table on page 14.

```
"←]←←←ST )Ø'3=ENO?>)B@/↑#9+G
```

REGRESSION ANALYSIS EXAMPLE -- BIG TEST

DATA

```
4A   4B   E
1.4103W  .54X  8Y  [
1.4099W  .55X  [
1.4094W  .56X  [
1.409 W  .57X  [
1.4086W  .58X  [
1.4081W  .59X  [
1.4077W  .60X  [
1.4072W  .61X  [
1.4068W  .62X  [

1.4376W  .54X  8.5Y  [
1.4372W  .55X  [
1.4368W  .56X  [
1.4364W  .57X  [
1.4360W  .58X  [
1.4356W  .59X  [
1.4352W  .60X  [
1.4348W  .61X  [
1.4344W  .62X  [

1.4646W  .54X  9Y  [
1.4643W  .55X  [
1.4639W  .56X  [
1.4636W  .57X  [
1.4632W  .58X  [
1.4629W  .59X  [
1.4625W  .60X  [
1.4622W  .61X  [
1.4618W  .62X  [

1.4915W  .54X  9.5Y  [
1.4912W  .55X  [
1.4909W  .56X  [
1.4906W  .57X  [
1.4903W  .58X  [
1.4900W  .59X  [
1.4897W  .60X  [
1.4893W  .61X  [
1.4890W  .62X  [

1.5185W  .54X  10Y  [
1.5182W  .55X  [
1.5179W  .56X  [
1.5176W  .57X  [
1.5173W  .58X  [
1.5170W  .59X  [
1.5167W  .60X  [
1.5165W  .61X  [
1.5162W  .62X  [
```

PARAMETERS

```
7760 7774 7774 7777 0000 0060
```

REGRESSION COEFFICIENTS

```
 2.030484E+00 -7.667827E-01  1.211007E+00  9.551607E-01
-2.310170E-01 -2.370390E-01  6.261075E-01 -9.978330E-01
 2.599291E-02  5.195604E-02 -9.607082E-02  1.119160E-01
-1.128037E-03 -1.461558E-04 -1.507848E-03  9.501156E-05
```

RESULTS

4A 4B E	V W	(V - W) X	N Y	Z
1.4103W .54X 8Y [				
	1.410078E+00	-2.219826E-04	1.000000E+00	
	1.410300E+00	5.400000E-01	8.000000E+00	1.000000E+00
1.4099W .55X [				
	1.409951E+00	5.067885E-05	2.000000E+00	
	1.409900E+00	5.500000E-01	8.000000E+00	1.000000E+00
1.4094W .56X [				
	1.409744E+00	3.439337E-04	3.000000E+00	
	1.409400E+00	5.600000E-01	8.000000E+00	1.000000E+00
1.409 W .57X [				
	1.409459E+00	4.587770E-04	4.000000E+00	
	1.409000E+00	5.700000E-01	8.000000E+00	1.000000E+00
1.4086W .58X [				
	1.409096E+00	4.963577E-04	5.000000E+00	
	1.408600E+00	5.800000E-01	8.000000E+00	1.000000E+00
1.4081W .59X [				
	1.408658E+00	5.576760E-04	6.000000E+00	
	1.408100E+00	5.900000E-01	8.000000E+00	1.000000E+00
1.4077W .60X [				
	1.408144E+00	4.439652E-04	7.000000E+00	
	1.407700E+00	6.000000E-01	8.000000E+00	1.000000E+00
1.4072W .61X [				
	1.407556E+00	3.562570E-04	8.000000E+00	
	1.407200E+00	6.100000E-01	8.000000E+00	1.000000E+00
1.4068W .62X [				
	1.406896E+00	9.568036E-05	9.000000E+00	
	1.406800E+00	6.200000E-01	8.000000E+00	1.000000E+00
1.4376W .54X 8.5Y [				
	1.436834E+00	-7.663816E-04	1.000000E+01	
	1.437600E+00	5.400000E-01	8.500000E+00	1.000000E+00
1.4372W .55X [				
	1.436621E+00	-5.789101E-04	1.100000E+01	
	1.437200E+00	5.500000E-01	8.500000E+00	1.000000E+00
1.4368W .56X [				
	1.436346E+00	-4.543066E-04	1.200000E+01	
	1.436800E+00	5.600000E-01	8.500000E+00	1.000000E+00
1.4364W .57X [				
	1.436011E+00	-3.890395E-04	1.300000E+01	
	1.436400E+00	5.700000E-01	8.500000E+00	1.000000E+00
1.4360W .58X [				
	1.435621E+00	-3.791600E-04	1.400000E+01	
	1.436000E+00	5.800000E-01	8.500000E+00	1.000000E+00
1.4356W .59X [				
	1.435179E+00	-4.213601E-04	1.500000E+01	
	1.435600E+00	5.900000E-01	8.500000E+00	1.000000E+00
1.4352W .60X [				
	1.434689E+00	-5.114824E-04	1.600000E+01	
	1.435200E+00	6.000000E-01	8.500000E+00	1.000000E+00
1.4348W .61X [				
	1.434154E+00	-6.460995E-04	1.700000E+01	
	1.434800E+00	6.100000E-01	8.500000E+00	1.000000E+00
1.4344W .62X [				
	1.433579E+00	-8.213520E-04	1.800000E+01	
	1.434400E+00	6.200000E-01	8.500000E+00	1.000000E+00

```
1.4646W  .54X   9Y  [
 1.464614E+00   1.408160E-05   1.900000E+01
 1.464600E+00   5.400000E-01   9.000000E+00   1.000000E+00
1.4643W  .55X   [
 1.464334E+00   3.378093E-05   2.000000E+01
 1.464300E+00   5.500000E-01   9.000000E+00   1.000000E+00
1.4639W  .56X   [
 1.464012E+00   1.124889E-04   2.100000E+01
 1.463900E+00   5.600000E-01   9.000000E+00   1.000000E+00
1.4636W  .57X   [
 1.463657E+00   5.693734E-05   2.200000E+01
 1.463600E+00   5.700000E-01   9.000000E+00   1.000000E+00
1.4632W  .58X   [
 1.463274E+00   7.365644E-05   2.300000E+01
 1.463200E+00   5.800000E-01   9.000000E+00   1.000000E+00
1.4629W  .59X   [
 1.462869E+00  -3.077090E-05   2.400000E+01
 1.462900E+00   5.900000E-01   9.000000E+00   1.000000E+00
1.4625W  .60X   [
 1.462451E+00  -4.942715E-05   2.500000E+01
 1.462500E+00   6.000000E-01   9.000000E+00   1.000000E+00
1.4622W  .61X   [
 1.462024E+00  -1.758188E-04   2.600000E+01
 1.462200E+00   6.100000E-01   9.000000E+00   1.000000E+00
1.4618W  .62X   [
 1.461597E+00  -2.032965E-04   2.700000E+01
 1.461800E+00   6.200000E-01   9.000000E+00   1.000000E+00

1.4915W  .54X   9.5Y  [
 1.492196E+00   6.956160E-04   2.800000E+01
 1.491500E+00   5.400000E-01   9.500000E+00   1.000000E+00
1.4912W  .55X   [
 1.491852E+00   6.521046E-04   2.900000E+01
 1.491200E+00   5.500000E-01   9.500000E+00   1.000000E+00
1.4909W  .56X   [
 1.491495E+00   5.948693E-04   3.000000E+01
 1.490900E+00   5.600000E-01   9.500000E+00   1.000000E+00
1.4906W  .57X   [
 1.491134E+00   5.337894E-04   3.100000E+01
 1.490600E+00   5.700000E-01   9.500000E+00   1.000000E+00
1.4903W  .58X   [
 1.490779E+00   4.787594E-04   3.200000E+01
 1.490300E+00   5.800000E-01   9.500000E+00   1.000000E+00
1.4900W  .59X   [
 1.490440E+00   4.396588E-04   3.300000E+01
 1.490000E+00   5.900000E-01   9.500000E+00   1.000000E+00
1.4897W  .60X   [
 1.490127E+00   4.266351E-04   3.400000E+01
 1.489700E+00   6.000000E-01   9.500000E+00   1.000000E+00
1.4893W  .61X   [
 1.489850E+00   5.495101E-04   3.500000E+01
 1.489300E+00   6.100000E-01   9.500000E+00   1.000000E+00
1.4890W  .62X   [
 1.489618E+00   6.181896E-04   3.600000E+01
 1.489000E+00   6.200000E-01   9.500000E+00   1.000000E+00

1.5185W  .54X   10Y  [
 1.518354E+00  -1.456291E-04   3.700000E+01
 1.518500E+00   5.400000E-01   1.000000E+01   1.000000E+00
1.5182W  .55X   [
```

```
 1.517940E+00 -2.604276E-04  3.800000E+01
 1.518200E+00  5.500000E-01  1.000000E+01  1.000000E+00
1,5179W  .56X  [
 1.517543E+00 -3.566593E-04  3.900000E+01
 1.517900E+00  5.600000E-01  1.000000E+01  1.000000E+00
1,5176W  .57X  [
 1.517179E+00 -4.210770E-04  4.000000E+01
 1.517600E+00  5.700000E-01  1.000000E+01  1.000000E+00
1,5173W  .58X  [
 1.516860E+00 -4.399270E-04  4.100000E+01
 1.517300E+00  5.800000E-01  1.000000E+01  1.000000E+00
1,5170W  .59X  [
 1.516600E+00 -3.997237E-04  4.200000E+01
 1.517000E+00  5.900000E-01  1.000000E+01  1.000000E+00
1,5167W  .60X  [
 1.516413E+00 -2.867729E-04  4.300000E+01
 1.516700E+00  6.000000E-01  1.000000E+01  1.000000E+00
1,5165W  .61X  [
 1.516312E+00 -1.876354E-04  4.400000E+01
 1.516500E+00  6.100000E-01  1.000000E+01  1.000000E+00
1,5162W  .62X  [
 1.516311E+00  1.114309E-04  4.500000E+01
 1.516200E+00  6.200000E-01  1.000000E+01  1.000000E+00
```

RMS or Standard Deviation

4.226363E-04

DRC = Dump Regression Coefficients. QUICK LOAD will put these coefficients in your program.

BT = Big Test. Put in switch register before start at 4420. See translate table on page 14.

```
$←\←\←←BT P04#YN;P&(E[V0,V!)00'I"M+QW9'VGQWE=+%0% D2?P'↑X←>/>IW
BC/FIG[90N)@8\/O*:'KN\GM46NDR@M(N↑+1>←.>.@>+
```

ADDRESS SUMMARY

27 BIT FLOATING POINT PACKAGE
5200-6777

PROG.	FIX	ANTL	RINT	FIN	LOG	FOUT	EDIT
ADDR.	5234	5271	6502	6600	6661	6200	5200

					--------HALTS---------			
PROG.	TYPE	C.R.	POA	FPNT	EXP.	EXP.	DIV.	
ADDR.	6305	5066	5110	5400	OVER	UNDER	BY	NEG.
INST.	4475	4474	4473	4476	FLOW	FLOW	ZERO	LOG
					5525	6046	6064	7402

TRACE ROUTINE
5034--5177

		---TRACE----			
	LINK	LIVE	DEAD		
INSERT	5134	5477	2200	LOW	HIGH
@ ADDR.	0077	5404	5404	5050	5051

TEMPORARY STORAGE ON PAGE ZERO

	Z					PROG. LOG
VAR.	Y	F	INT	CNT	CH	ANTL
ADDR.	35-37	40-42	43-45	46	47	

FPNT	--SIGN----		-EXPONENT-			
VAR.	OP	AC	OP	AC	AC 3-1	OP 3-1
ADDR.	50	51	52	61	53-55	56-60