original

A Collection of Information on The TI-74 and CC-40 Computers

by

Palmer O. Hanson, Jr.
Editor - TI PPC Notes

July 1988

This collection is a compilation of articles on the TI-74 and CC-40 and peripherals which appeared in the Volume 12 (1987-1988) issues of TI PPC Notes.

REDUCED TI-74 AND TI-95 PRICES - V12N2P4 listed prices from various suppliers. A recent visit to the local Service Merchandise outlets shows that they have reduced prices for TI-74 and TI-95 hardware by some ten to eighteen percent. The current prices are:

TI-74	$ 99.97	ROM Modules	$29.80
TI-95	129.90		

TI PPC NOTES V12N1P9

MORE ON A DISK DRIVE FOR THE TI-74 - I received a flyer on the Mechatronic Quickdisk-02 Drive from Technical Application Product Engineering, Ltd., 1439 Solano Place, Ontario, California 91764. Storage capacity is listed as 64K on each side. Operation is possible with either a power supply or AA batteries. No prices are given.

TI PPC NOTES V12N3P15

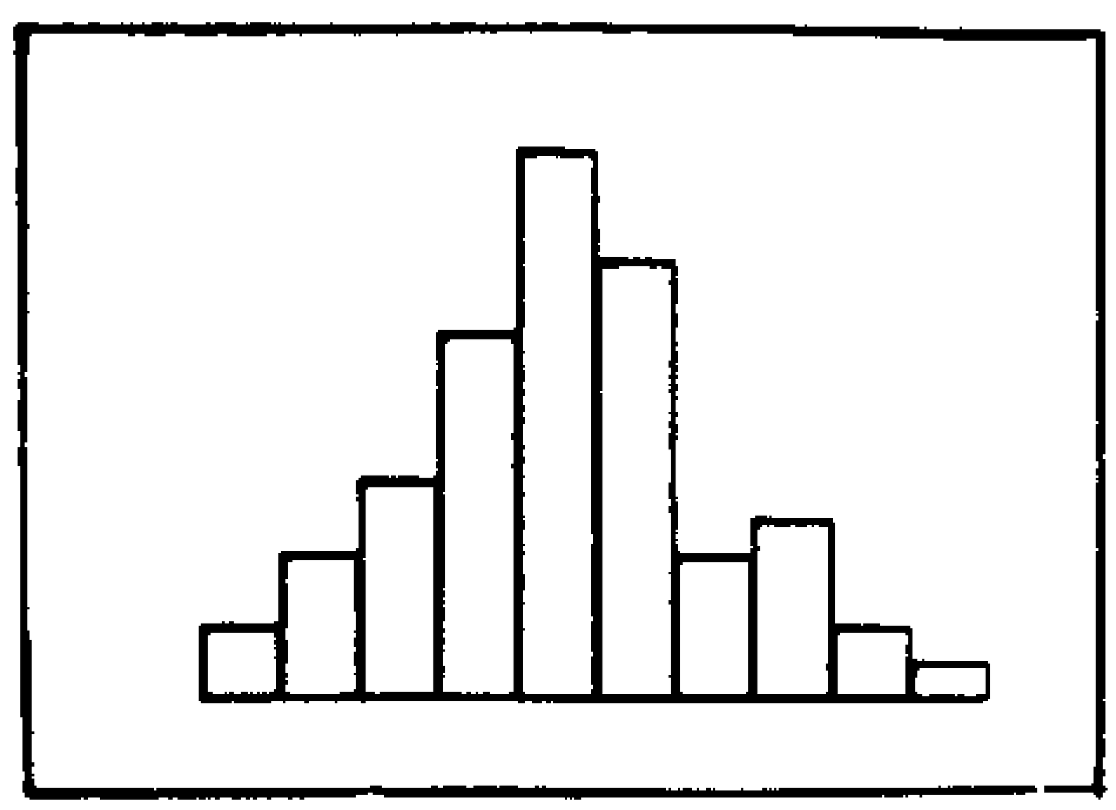
BAR GRAPHS AND HISTOGRAMS

A routine for the TI-74 and PC-324 is not complex due to the availability of the RPT$ command. See the program at the right. To use the plot routine the values to be plotted must be stored in sequence in the Array U. Lines 1020 through 1040 of the routine use Maurice Swinnen's technique for entering values as strings, and for marking the end of input with entry of an "E". The routine in lines 1050 through 1070 print the bar graph, where line 1060 uses the RPT$ command to generate a sequence of O's of the required length. The bar graph is the same as that from the TI-95 and PC-324.

```
1000 DIM U(20)
1010 OPEN #1,"12",OUTPUT
1020 INPUT X$
1030 IF X$="E"OR X$="e"T
HEN 1050
1040 K=K+1:U(K)=VAL(X$):
GOTO 1020
1050 FOR I=1 TO K
1060 PRINT #1,RPT$("O",U
(I))
1070 NEXT I
1080 END
```

For more elegant bar graphs the TI-74 user can use the interface cable from page 13 of this issue to obtain the graphics capability of the HX -1000 Printer/Plotter. A sample program and bar graph appear below.

```
100 DIM X(20)
110 INPUT X$
120 IF X$="E"OR X$
="e"THEN 200
130 K=K+1
140 X(K)=VAL(X$)
150 GOTO 110
200 OPEN #1,"10",O
UTPUT
210 PRINT #1,CHR$(
19)
300 FOR I=1 TO K
310 Y$=STR$(X(I)*1
0)
320 P$="L(0,0),("&
Y$&",0),("&Y$&",-2
0),(0,-20),(0,0)"
330 PRINT #1,"O"
340 PRINT #1,P$
350 PRINT #1,"M(0,
-21)"
360 NEXT I
400 PRINT #1,CHR$(
17)
410 CLOSE #1:END
```



TI PROGRAMMABLE CALCULATOR NEWS - This is a new publication by Texas Instruments which appears to be the free newsletter promised by material received with my TI-95. The eight page first issue provides coverage of the TI-74 and TI-95, but no coverage of any of the earlier devices such as the TI-59 or CC-40. There is no mention of the additional peripherals described in the survey telephone call that I received earlier this year (see page 7 of this issue).

There is one apparent mistake in the first issue. The first column of page 6 states that "In the calculator mode, the TI-74 offers 70 scientific functions, alphanumeric messages, and 13-digit accuracy which spans a numeric range of +/- 9.9999999999999E127. ...". I did copy the nines correctly. There were 14 in all, 13 to the right of the decimal point. More correct descriptions of the numerical accuracy of the TI-74 appear in the TI-74 Programming Reference Guide: "The TI-74 uses a minimum of 13 digits to perform calculations. ..." from page A-32, and "The TI-74 uses radix-100 format for internal calculations ... ... Another benefit of this technique is 13, and sometimes 14, digits of internal precision." A more complete discussion of radix-100 arithmetic appeared in pages F-1 and F-2 of the CC-40 User's Guide and in Laurance Leeds' treatise "Numeric Representation in the TI-99/4 and CC-40" in V9N5P6/7.

TI's Direct Marketing Program Manager has offered to provide copies of the first issue for our members. If they arrive in time a copy will have been included with this issue TI PPC Notes. If they do not, the copy will be included with the next issue. You can get on their mailing list by writing to Programmable Caclulator News, P. O. Box 53, Lubbock, TX 79408. Please mention TI PPC Notes.

TI PPC NOTES V12N4P3

A NEW CLUB FOR THE TI-74 AND TI-95 - Thomas Coppens, who previously was the editor of the TISOFT newsletter for users of the TI-59 and TI-99/4, has announced the beginning of a newsletter and software exchange club for the TI-74 and TI-95. The organization is called SeTIc, which stands for Software exchange for Texas Instruments calculators. The newsletter is available in either French or Dutch. A one year subscription is fifteen dollars ($15.00).

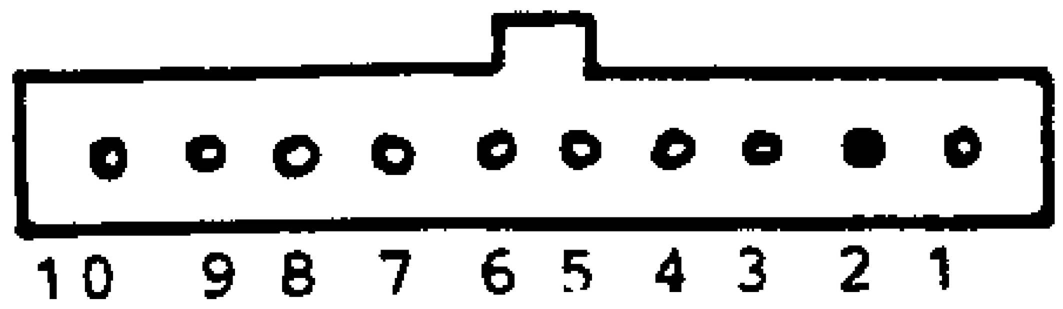
SeTIc has also published a program listing and flow chart for the Mathematics module of the TI-95. There is no explanatory material such as that which was in the so-called "Fish Book" foir the TI-59. The listing comes in a 6 inch by 8 inch loose leaf form. The price is ten dollars ($10.00) including shipping. This wil be a valuable book if you plan to use routines from the module in your programs, or if you want to analyze the routines in the module.

To order these materials send an international postal money order (no checks, please!) to:

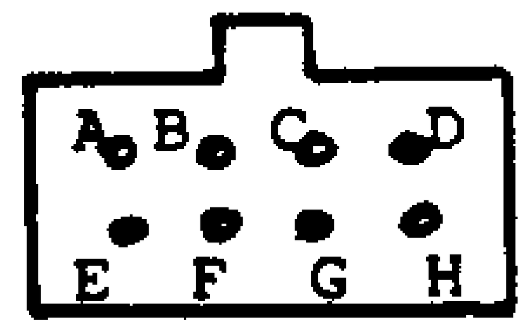
Thomas Coppens
P. O. Box 63
2080 Kapellen
Belgium

State whether you want the French or Dutch versions.

USING CC-40 PERIPHERALS WITH THE TI-74 - Maurice Swinnen



Rear view TI-74



Hex-Bus cable

In spite of reports to the contrary, it is possible to use all of the CC-40 peripherals with the TI-74. All you need is a special interconnecting cable. The connections have been kept a deep secret by certain interested parties. Experimenation revealed the following:

The TI-74 has a 10-pin connector as show above. Diameter of the pins is 0.016" or 0.4 mm. Distance between pins is 0.1" or 2.54 mm. Does that ring a bell? Of course it does! That is the same distance and diameter used with ICs and its inline dip-sockets. The connector on the TI-74 uses only 8 of the 10 pins, pins 1 through 8. To make a good female connector for it, simply saw off half of a 16-pin dip-socket, preferrable one used for wire-wrap, with long, sturdy wire-wrap stems. They allow for neat soldering. If you then shrink enough heat-shrinkable tubing around each stem so as to fill up the space between stems, it is possible to shrink one, large-diameter piece of heat-shrinkable tubing around everything and obtain an almost perfect socket.

If you prefer to have all ten pins used, cut one side of a 24-pin dip-socket and trim off two pins. That will give you a 10-pin socket. You could even glue a small hump on the top, to prevent reverse plug-in. (I tried reverse plug-in; it doesn't harm anything, but, of course, it doesn't work either)

For the hex-bus connector you will have to sacrifice a dual-plug hex-bus cable. Take a long one, cut it in two, and share the other half with a friend. When you strip the wires, you will find, of course, 8 of them. All will be soldered to corresponding pins on your TI-74 connector, except one, the green one. Insulate it, and forget it. Wires are soldered as follows:

TI-74 connector	Hex-Bus connector	Color
1	D	orange
2	not connected	
3	C	red
4	E	brown
5	H	blue
6	G	black
7	B	yellow
8	A	grey
9	not connected	
10	not connected	

I tried it with the following peripherals with great success: Printer 80, RS232 Interface, the Disk Drive and the Printer-Plotter. I tried to have one TI-74 to talk to one CC-40, without success. The TI-74 is willing, but the CC-40 acts finicky and tells me that its memory may be lost. Well...

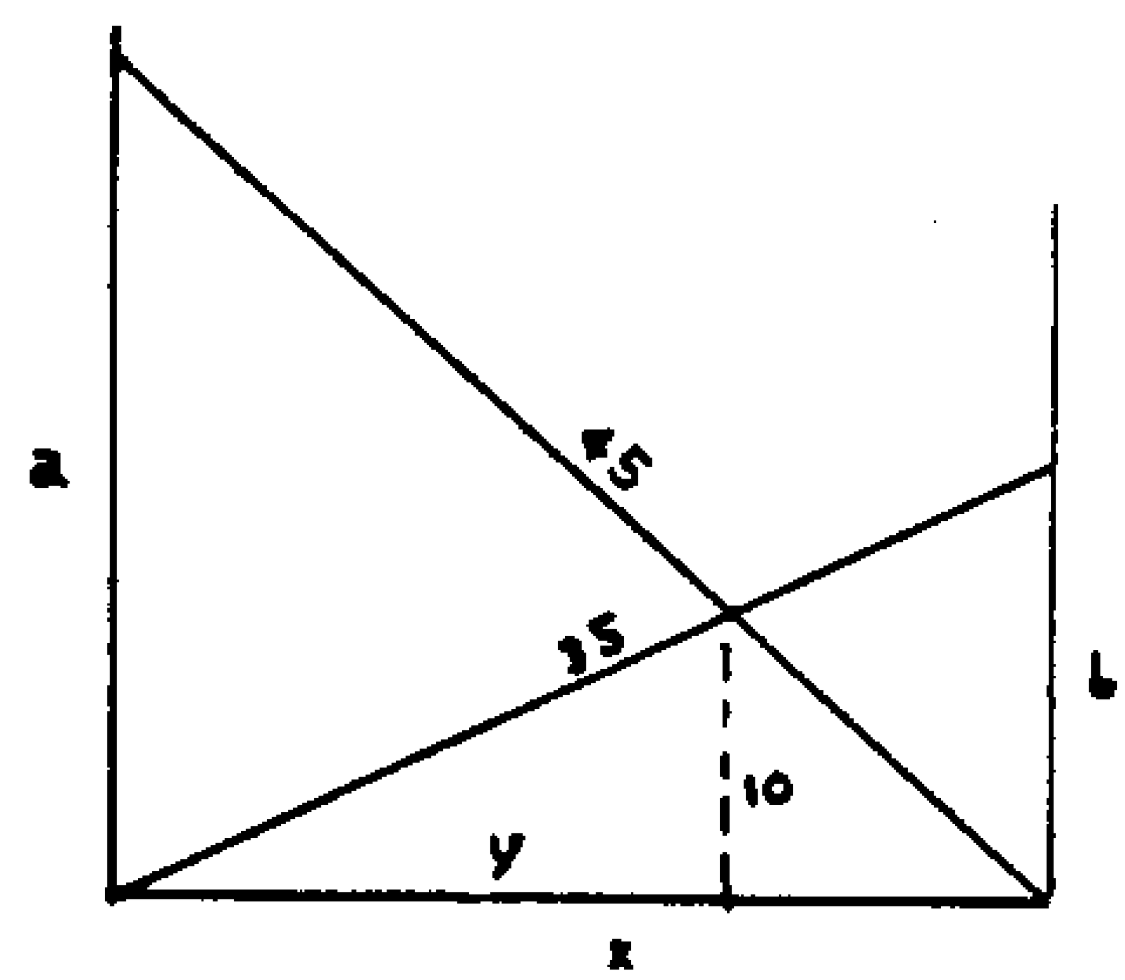
Editor's Note: I made the cable and successfully demonstrated use of My TI-74 with the HX-1000. This cabling has not been approved by TI and use could result in loss of warranty.

SOLVING LADDER PROBLEMS - P. Hanson. I have always been interested in the solution of the so-called "ladders in an alley" problems. My interest was rekindled by an article in a recent in-house Honeywell publication which reported the use a ladder problem to illustrate the problem-solving capability of Borland's "Eureka: The Solver" program.

There are two kinds of "ladders in the alley" problem. For both the ladders are set so that they touch the ground at one side of the alley and lean against the building on the other side of the alley. The lengths of the ladders are given. In the first problem the width of the alley is given, and the problem is to find the height at which the ladders cross. This is solvable in a straightforward manner as the intersection of two straight lines.

In the second "ladders in the alley" problem the height at which they cross is given. The problem is to find the width of the alley. The classical analysis eventually reduces the solution to a fourth degree polynomial in the square of the width. The capability of the "Eureka: The Solver" program to solve simultaneous nonlinear equations avoids the necessity to reduce the problem to a polynomial in x. The Mathematics Library module for the TI-95 has a similar capability, so this ladder problem provides a good vehicle to demonstrate the use of that routine, and to demonstrate other methods of solution.

Consider the ladder problem illustrated at the right, where one ladder is 35 feet long, the other ladder is 45 feet long, and the ladders cross 10 feet above the ground. The heights at which the ladders touch the opposite walls are designated a and b, the width of the alley is x, and the distance from one side to a point directly below the point at which the ladders cross is y. Then, four equations in a, b, x, and y may be written as:



(1) $x^2 + a^2 = 45^2$

(2) $x^2 + b^2 = 35^2$

(3) $y/10 = x/b$

(4) $(x-y)/10 = x/a$

The four equations in four unknowns above can be easily reduced to two equations in two unknowns by substituting equation (4) into equation (1) to remove a, and by substituting equation (3) into equation (2) to remove b. The resulting equations are:

(5) $y = x - 10x/\sqrt{(2025 - x^2)}$

(6) $y = 10x/\sqrt{(1225 - x^2)}$

Subtracting one of those equations from the other yields an new equation in x where the zero is the desired solution. That function can be solved with any "zeroes of a function" routine.

(7) $y = x - 10x(1/\sqrt{(2025 - x^2)} + 1/\sqrt{(1225 - x^2)})$

Solution using Zeroes of a Function

Solutions using zeroes of a function routines with equation (7) can be obtained with the fx-7000G using the technique described on V11N3P10, with the TI-58/59 using ML-08, and with the TI-74 or TI-95 using either the bisection or Newton-Raphson methods from the Mathematics Library modules. I will begin with the TI-74.

The bisection rootfinder program requires that the function be entered as a subprogram. The subprogram I used is illustrated at the right. For boundary values of A = 1 and B = 34.9, and Epsilon = 0.000001 the program calculates x = 31.81745886 in about 25 seconds. If the printer is used the printout at the right will occur. If you try to use A = 1 and B = 35 the program will stop with the error message "W26 at 1001 Division by Zero". If instead you try to use A = 1 and B = 36 the program will stop with the error message "E23 at 1001 Bad Argument". Both of those errors might be expected due to the SQR(1225 - x^2) in the denominator of the function.

```
1000 SUB FX(X,Y)
1001 Y=X*(1-10*(1/SQR(20
25-X*X)+1/SQR(1225-X*X))
)
1002 SUBEND
```

```
ROOT FINDER-BISECTION

A=
 1
B=
 34.9
Epsilon=
 .000001

Root=
 31.81745886
```

The Newton-Raphson root finder in the TI-74 Mathematics Library requires that both the function and its derivative be entered as subprograms. The subprograms I used are illustrated at the right. Note that I did not use the equation which is the exact derivative of the function, but rather used an approximation by finding the slope in the neighborhood of x by numerically evaluating $(f(x + \Delta x) - f(x))/\Delta x$ where Δx was selected as 0.0001. For an initial guess of Xo = 29 and Epsilon = 0.000001 the program finds x = 31.8174591 in about 10 seconds. If the printer is used the printout at the right will occur. If you try to use an Xo less than 28.57 the program will stop with the error message "E23 at 1001 Bad Argument". You might think that situation could be relieved by defining a better derivative function. However, if you run the Newton root finder in the TI-95 Mathematic Library, where a user defined derivative is not required, you will experience a similar problem where an Xo less than 28.57 results in the error message "INVALID ARGUMENT".

```
1000 SUB FX(X,Y)
1001 Y=X*(1-10*(1/SQR(20
25-X*X)+1/SQR(1225-X*X))
)
1002 SUBEND
1003 SUB FD(X,Y)
1004 Y=X*(1-10*(1/SQR(20
25-X*X)+1/SQR(1225-X*X))
)
1005 A=X+.0001
1006 B=A*(1-10*(1/SQR(20
25-A*A)+1/SQR(1225-A*A))
)
1007 Y=(B-Y)/.0001
1008 SUBEND
```

```
ROOT FINDER-NEWTON

Xo=
 29
Epsilon=
 .000001
Root=
 31.8174591
```

In both the TI-74 and TI-95 an Xo between about 19.4 and 28.57 will stop the program. A Xo between 0 and 19.3 will find the root at zero. There is also an inconsistency in documentation for the program. Page 3-16 of the TI-74 Mathematics Library Guidebook indicates that approximations are used by the program to compute the derivative. The approximation formulas presented are similar in concept to the ones that I used. Now, if the program implements a derivative internally, then why must I enter a subprogram to compute the derivative? I thought that I might not need to have a subprogram FD, but when I try to run without one I get an error message "E13 Not Found". Can anyone explain?

CURVE FITTING WITH ORTHOGONAL POLYNOMIAL METHODS

Pages 155 through 160 of the June 1987 issue of BYTE presented a BASIC program by William Hood which uses orthogonal polynomial methods for curve fitting. The program as listed in BYTE is not compatible with either the CC-40 or the TI-74. The listing at the right is compatible. Mr. Hood has kindly given us permission to present the modified program. He asks that any inquiries about the modified program be addressed to TI PPC Notes. The changes which were made relative to the listing in BYTE were:

Lines 1000-1050 were changed to accept the peculiarities of the TI-74 such as

Variable Dimensioning not allowed in the TI-74.

Functions are not available on the TI-74.

LN changed to LQ since LN is a reserved word in the TI-74.

Deletion of lines 1020 and 1030 since the TI-74 sets all variables to zero at program entry.

Lines 1030-1045 provide printer control needed by the TI-74 and PC-324.

Line 1050 establishes a printout format to be used by the PRINT USING command at line 1780.

Line 1060 has a change in the GOTO address.

Lines 1070-1310 are the same as in the BYTE listing. Lines 1320-1550 of the BYTE program were deleted to conserve memory.

Lines 1400-1550 provide a data entry routine more consistent with the capabilities of the TI-74. A prompt is provided for the entry of each X, Y, or W value.

Line 1680 mechanizes the same equation as in the BYTE listing, but without the use of a function. The solution illustrates a little-used feature of BASIC, that a relational expression evaluates to -1 if the condition is true or to 0 if the condition is false. See page 1-12 of the TI-74 Programming Reference Guide.

Lines 1690-1710 are the same as in the BYTE listing.

Lines 1720-1750 provide an altered output of the polynomial coefficients to implement a preference for C(0) as the constant, C(1) as the coefficient of the first degree term, etc.

```
1000     dim X(50),Y(50)
,W(50),C(11)
1010     DIM D1(11),D2(1
1),D3(11),D4(11),D5(11),
D6(11)
1020 LQ=50:LD=11
1030 INPUT "Use Printer
(Y/N)? ";A$
1035 IF A$="Y"OR A$="y"T
HEN PN=1
1040 IF PN=1 THEN INPUT
"Enter device name: ";F$
1045 IF PN=1 THEN OPEN #
PN,F$,OUTPUT
1050 IMAGE #.######
1060 GOTO 1400
1070 IF MF<0 AND M<MM TH
EN J1=MM+1:MM=M:GOTO 113
0
1080 J1=1:MM=M:S1=0:S2=0
:S3=0:S4=0
1090 FOR I=1 TO N:WT=W(I
)
1100 S1=S1+WT*X(I):S2=S2
+WT:S3=S3+WT*Y(I):S4=S4+
WT*Y(I)*Y(I)
1110 NEXT I
1120 D4(1)=S1/S2:D5(1)=0
:D6(1)=S3/S2:D1(1)=0:D2(
1)=1:VR=S4-S3*D6(1)
1130 FOR J=J1 TO MM:S1=0
:S2=0:S3=0:S4=0
1140 FOR I=1 TO N:P1=0:P
2=1
1150 FOR K=1 TO J:P=P2:P
2=(X(I)-D4(K))*P2-D5(K)*
P1:P1=P:NEXT K
1160 WT=W(I):P=WT*P2*P2
1170 S1=S1+P*X(I):S2=S2+
P:S3=S3+WT*P1*P1:S4=S4+W
T*Y(I)*P2:NEXT I
1180 D4(J+1)=S1/S2:D5(J+
1)=S2/S3:D6(J+1)=S4/S2:D
3(1)=-D4(J)*D2(1)-D5(J)*
D1(1)
1190 IF J<4 THEN 1210
1200 FOR K=2 TO J-2:D3(K
)=D2(K-1)-D4(J)*D2(K)-D5
(J)*D1(K):NEXT K
1210 IF J>2 THEN D3(J-1)
=D2(J-2)-D4(J)*D2(J-1)-D
5(J)
1220 IF J>1 THEN D3(J)=D
2(J-1)-D4(J)
1230 FOR K=1 TO J:D1(K)=
D2(K):D2(K)=D3(K):D6(K)=
D6(K)+D3(K)*D6(J+1):NEXT
 K
1240 NEXT J
1250 FOR J=1 TO M+1:C(J)
=D6(M+2-J):NEXT J
1260 P2=0:FOR I=1 TO N:P
=C(1)
1270 FOR J=1 TO M:P=P*X(
I)+C(J+1):NEXT J
1280 P=P-Y(I):P2=P2+W(I)
*P*P:NEXT I
```

Curve Fitting with Orthogonal Polynomials - (cont)

Lines 1760-1770 are the equivalent of line 1750 in the BYTE listing.

Lines 1775-1790 are the equivalent of lines 1760-1770 in BYTE. The PRINT USING command at line 1780 together with the IMAGE command at line 1050 accomplish the same limiting of the answer to six decimal places as line 1760 of the BYTE listing.

Lines 1800-1870 provide a printout of the residual errors. I have found that this is more useful in finding bad data entry than the tables of the input and calculated dependent variables.

Lines 1900-1910 permit the solution for another degree. I moved this downstream from the table of residuals. This permits reviewing the residuals before selecting another degree. The BYTE program provides the option for another solution before the printout of the data. That means that the user cannot get a printout of the data if he also want to try another degree.

Lines 1900-2130 of the BYTE program were deleted.

The program permits use of weighted or unweighted data. A complete set of prompts are available. With the PC-324 the response to the "Enter Device Name:" prompt is 12.

The run time for the solution to the 10 data pair problem outlined in the BYTE article is very similar that for the row reduction program on page 14 of this issue, and about a factor of two faster than the V11N4P12 solution using in the Mathematics module for the TI-74. The sums of the squares of the residuals are equal to nine decimal places for all three programs. A printout for the sample problem from BYTE appears below.

```
C(0) = -.5700247169
C(1) =  7.19996658
C(2) = -1.113133706
C(3) =  .0669397883
C(4) = -.0012073187

Residual Variance =
 .9917402194

Coeff of Det (R^2) =
 .984184
```

```
D(1) = -.2400328374
D(2) =  .9451840484
D(3) = -1.028420627
D(4) =  .8898320409
D(5) = -.571941066
D(6) =  .4281552822
D(7) = -.8772289335
D(8) =  .7686350189
D(9) = -.502158865
D(10) =  .1879759383
```

```
1290 S2=0:IF N>M+1 THEN
S2=P2/(N-M-1)
1300 R2=1:IF VR<>0 THEN
R2=1-P2/VR:IF R2<0 THEN
R2=0
1310 RETURN
1400 INPUT "Is the data
weighted (Y/N)? ";W$
1410 IF W$<>"Y"AND W$<>"
y"THEN W$="N"
1420 INPUT "How Many Dat
a Points? ";N
1430 IF N<2 OR N>LQ THEN
 1420
1500 FOR K=1 TO N
1510 INPUT "X("&STR$(K)&
") = ";X(K)
1520 INPUT "Y("&STR$(K)&
") = ";Y(K)
1530 IF W$="N"THEN W(K)=
1:GOTO 1550
1540 INPUT "W("&STR$(K)&
") = ";W(K)
1550 NEXT K
1680 PM=((LD-1)<(N-1))*(
1-LD)+((N-1)<=(LD-1))*(1
-N)
1690 INPUT "Degree of Po
lynomial? ";M
1700 IF M<1 OR M>PM THEN
 1690
1710 GOSUB 1070
1720 FOR J=M+1 TO 1 STEP
 -1
1725 PRINT #PN,"C("&STR$
(M+1-J)&") = ";C(J)
1730 IF PN=0 THEN PAUSE
1740 NEXT J
1750 PRINT #PN
1760 PRINT #PN,"Residual
 Variance =     ";S2
1765 IF PN=0 THEN PAUSE
1770 PRINT #PN
1775 PRINT #PN,"Coeff of
 Det (R^2) = "
1780 PRINT #PN,USING 105
0,R2
1785 IF PN=0 THEN PAUSE
1790 PRINT #PN
1800 INPUT "Print the re
siduals (Y/N)? ";A$
1810 IF A$<>"Y"AND A$<>"
y"THEN 1900
1820 FOR I=1 TO N:P=C(1)
1830 FOR K=1 TO M:P=P*X(
I)+C(K+1):NEXT K
1840 PRINT #PN,"D("&STR$
(I)&") = ";P-Y(I)
1850 IF PN=0 THEN PAUSE
1860 NEXT I
1870 PRINT #PN
1900 INPUT "Try another
degree (Y/N)? ";A$
1910 IF A$="Y"OR A$="y"T
HEN 1680
3000 END
```

TI PPC NOTES V12N2P2

ERRATA

Curve Fitting with Orthogonal Polynomials (V12N1P24) - George Thomson found that the following changes are needed in the TI-74 version of William Hood's program. First, add a new line at 1778. Then, change the <'s to >'s in line 1070, and change line 1910 to set MF = 1. The revised lines will be:

```
1070 IF MF>0 AND M>MM THEN J1=MM+1:MM=M:GOTO 1130
1778 IF PN=0 THEN PAUSE 1
1910 IF A$="Y" OR A$="y" THEN MF=1:GOTO 1680
```

The change at line 1778 flashes "Coeff of Det (R^2) = " before stopping with the value in the display when a printer is not used. The changes at lines 1070 and 1910 provide a faster solution for a polynomial of a higher degree if a solution for a polynomial of lower degree has already been completed. Without those changes the fitting a third degree polynomial to the ten point problem from BYTE requires about 13 seconds and the fitting a fourth degree polynomial to the problem requirs an additional 17 seconds. With the changes the third degree solution would still require about 13 seconds if it were the first solution, but the fourth degree solution can be obtained in an additional eight seconds.

TI PPC NOTES V12N2P7

A PROGRAMMING CHALLENGE - While faster algorithms have been know to be available, (e.g., see The Art of Computer Programming by D. Knuth, volume 2, pp 364-398), the factor finders in previous issues of TI PPC Notes have all been versions of the sieve of Eratosthenes. The notice at the right, which is from Page 59 of issue #35 of the EduCALC catalog, offers discussions and examples for one of the faster algorithms. Write to Algorithms Dept., EduCALC Mail Store, 27953 Cabot Road, Laguna Niguel, CA 92677 for a copy. Remember to include the self-addressed envelope with 39 cents in stamps.

Of course, the challenge is to show that more than five or six digits can be obtained from TI Programmables.

Factoring Large Numbers on the HP-16C

Factoring large numbers intrigues both amateur and serious number theorists and factoring now gets increasing attention with recent applications to cryptography. You may be interested in the factoring algorithms that are offered in a recent article by Blair/Lacampagne/Selfridge.

In addition to a short discussion and a step-by-step description of each algorithm, they include programs that factor numbers up to 19 digits, fast, using the HP-16C! Interestingly, these algorithms would only work up to about 5 or 6 digit numbers on other calculators.

As a public service, EduCALC will send you a copy of this article if you send us a self-addressed envelope with 39 cents in postage on it—please send it to our "Algorithms Dept".

CC-40 SOLID STATE CARTRIDGES FOR SALE - The cartridges which are available include a mathematics cartridge, a memo processor/ data communications cartridge, a 16K RAM cartridge, a battery backed-up RAM cartridge, and an editor/assembler cartridge. Write to David R. Hertling, 4546 Cherie Glen Trail, Stone Mountain, GA 30083.

TI PPC NOTES V12N1P2

CC-40 For Sale - Write to Arthur O. Jacobsen, 56 Maguire avenue, Avon, MA 02322.

MORE USEFUL FUNCTIONS ON THE CC-40 AND TI-74 - P. Hanson. Page 5 describes my discovery that the CC-40 and TI-74 permit algebraic expressions as the response to an input statement. I decided to look through the manuals for additional capabilities that I may have missed. I found the PAUSE ALL statement and the subprogram capabilities, both of which provide easier solutions for certain programming requirements.

The PAUSE ALL statement suspends program execution each time a complete output line has been sent to the display, and execution continues when the CLR or ENTER key is pressed. When used with PRINT #PN statements where PN is set to zero if the printer is not connected, or set to one if the printer is connected, the PAUSE ALL statement provides an easy way to stop execution with a result in the display if a printer is not used, but to continue without stoppping if a printer is used. The alternative which I used in earlier programs was a statement such as IF PN = 0 THEN PAUSE each time the option was desired.

The subprogram capability is useful when the programmer would like to use the same mathematical routine several places in the same program, but with different definitions for the variables. The argument list in the CALL statement passes a variable list from the main program to the subprogram. The SUB statement which marks the beginning of a subprogram can redefine the variable list as desired. The subprogram capability is particularly useful in providing portability of a routine from one program to another.

Both the PAUSE ALL and the subprogram capability are demonstrated in the cubic program by Larry Leeds on page 10. The subprogram capability is also demonstrated in an iterative least squares program elsewhere in this issue, and in the DMS to decimal degrees program on page 7 of the first issue of Programmable Calculator News.

TI PPC NOTES V12N2P5

EXPRESSIONS AS THE RESPONSE THE INPUT STATEMENT WITH THE CC-40 AND TI-74 - P. Hanson

During evaluation of solutions for cubic equations we used a test problem proposed by Peter Messer: x - 2x + (4/3)x - 2/9 = 0. The fractional coefficients did not cause any difficulty when we were using calculator programs, but when Larry Leeds developed a BASIC program on his Model 100 he found that it was not so easy to enter the fractional coefficients, and he used program statements to enter the coefficients. When I converted his Model 100 program for use on the CC-40 and TI-74 I received a pleasant surprise. The Input statement with those machines permits a response with a numeric expression. An example will take the place of many words. Consider the program at the right. Press RUN and see a question mark in the display. Place 2/9 in the display and press ENTER. The value .2222222222 is printed. A question mark appears in the display indicating that a value for B will be accepted. Place SIN(PI/6) in the display and press ENTER. The value .5 is printed. A question mark in the display indicates that a value for C will be accepted. Place 2 + SQR(5) in the display and press ENTER. The value 4.236067978 is printed. A question mark indicates that a value for D will be accepted. Place A/B + C in the display and press ENTER. The value 4.680512422 is printed.

```
10 OPEN #1,"12",OUTPUT
20 RAD:INPUT A
30 PRINT #1,A
40 INPUT B
50 PRINT #1,B
60 INPUT C
70 PRINT #1,C
80 INPUT D
90 PRINT #1,D
99 END

.2222222222
.5
4.236067978
4.680512422
```

EVALUATION OF POLYNOMIAL CURVE FITTING PROGRAMS - V12N1P24/25 presented a program by William Hood for curve fitting using orthogonal polynomial techniques. A primary reason for using such techniques is reduced susceptibility to ill-conditioning; however, the sample provided with Hood's article in BYTE did not illustrate that effect. V12N2P25 reported that for the sample problem the sums of the squares of the residuals were equal to nine decimal places for the orthogonal polynomial method, for the row reduction program on V12N1P14 and for the V11N4P12 solution using the Math module of the TI-74.

One question is what are the correct coefficients of the solution. Richard Spurrior calculated the solution in quadratic precision using a version of Hood's program. The first twenty digits of his results for solutions with second through fourth degree polynomials are (constant terms first):

Degree	2	3	4
A0	5.3993800361687283825	0.4971695587517259304	-0.5700247169557546353
A1	1.5004168045476485057	5.7929781587696064511	7.1999665801335072608
A2	-0.0326693617568390723	-0.7139797262710357798	-1.1131337063816293808
A3		0.0281457952959845654	0.0669397883174221636
A4			-0.0012073187307998324
Variance	7.5798137801289391680	1.0206238865340448785	0.9917402194039777401
R-squared	0.8307672547295986789	0.9804680860539975219	0.9841840328617498167

George Thomson proposed that the so-called "Wampler quintics" would provide better insight into the relative capabilities of various curve fitting programs. The method was described in Roy H. Wampler's paper "An Evaluation of Linear Least Squares Computer Programs" in the April-June 1969 issue of the Journal of Research of the National Bureau of Standards. Two test polynomials were proposed:

$$y = 1 + x + x^2 + x^3 + x^4 + x^5$$

$$y = 1 + x/10 + (x/10)^2 + (x/10)^3 + (x/10)^4 + (x/10)^5$$

where 21 input data pairs are defined for values of x from 0 to 20. Thus, for the first quintic the y values range from 1 to 3368421, and for the second quintic the y values range from 1 to 63. The paper also defined a measure (C) of the number of correct decimal digits in a coefficient of the fitted polynomial as the negative log of the magnitude of the relative error. If a coefficient is exact the number of digits with which the machine computes is used. The paper uses the average of the C values for the six coefficients as the figure of merit of the solution. George Thomson prefers the use of the minimum C value as a figure of merit. Others like to consider the standard error of the residuals as well.

The following table compares various figures of merit for the first quintic for four programs which have been published in previous issues of TI PPC Notes:

Machine	TI-59	TI-74	TI-74	TI-74
Program	V9N2P20	V11N4P12	V12N1P14	V12N1P24
Method	Mat Inv	Mat Inv	Row Red	Orth Poly
C(ave)	3.65	4.36	5.62	7.41
C(min)	2.16	2.62	4.15	6.05
S.E.	1.95E-03	2.54E-01	2.26E-05	2.95E-06

Evaluation of Polynomial Curve Fitting - (cont)

Preliminary tests showed that the second quintic, the one with tens in the denominators, did not provide as much differentiation between the results from the different programs. Accordingly, the results for the second quintic have not been presented here.

Examination of the figures of merit on page 16 reveals one anomaly: the number of correct digits in the coefficients is slightly higher for the TI-74 matrix inversion routine than for the TI-59 matrix inversion program; however, the standard error of the residuals for the TI-59 program is two orders of magnitude smaller than for the TI-74 program. Examination of the details of the solutions for the four programs shows why:

A1 =	1.003396496
A2 =	0.9932158788
A3 =	1.002457193
A4 =	0.9996749308
A5 =	1.000017923
A6 =	0.9999996501
d1 =	.0033964957
d2 =	-.0012379293
d3 =	-.002667963
d4 =	-.0022512867
d5 =	-.000999407
d6 =	.000380358
d7 =	.001442883
d8 =	.00196327
d9 =	.00189487
d10 =	.00132724
d11 =	.000444
d12 =	-.0005187
d13 =	-.0013138
d14 =	-.0017271
d15 =	-.0016179
d16 =	-.0009602
d17 =	.000112
d18 =	.00127
d19 =	.001936
d20 =	.001256
d21 =	-.001952
Mean =	.000008425
S.E. =	.001952144

V9N2P20

A1 =	.9989
A2 =	1.0024
A3 =	.9996
A4 =	1.000067
A5 =	.999999
A6 =	1.0000001
d1 =	.0011
d2 =	-.0009661
d3 =	-.0026232
d4 =	-.0042523
d5 =	-.0062344
d6 =	-.0089625
d7 =	-.0128536
d8 =	-.0183607
d9 =	-.0259848
d10 =	-.0362869
d11 =	-.0499
d12 =	-.0675411
d13 =	-.0900232
d14 =	-.1182673
d15 =	-.1533144
d16 =	-.1963375
d17 =	-.248654
d18 =	-.311736
d19 =	-.387225
d20 =	-.476942
d21 =	-.5829
Mean Error =	-.1271938636
S.E. =	.2543009854

V11N4P12

A1 =	1.000030632
A2 =	.99992933
A3 =	1.000027049
A4 =	.9999963001
A5 =	1.000000209
A6 =	.9999999959
d1 =	-.000030632
d2 =	.0000164843
d3 =	2.890386E-05
d4 =	.0000219328
d5 =	.0000068624
d6 =	-.0000085344
d7 =	-.0000195047
d8 =	-.00002381
d9 =	-.00002128
d10 =	-.00001319
d11 =	-.00000192
d12 =	.00000965
d13 =	.00001857
d14 =	.00002241
d15 =	.00001966
d16 =	.00001026
d17 =	-.000004
d18 =	-.000019
d19 =	-.000026
d20 =	-.000016
d21 =	.000027
Mean =	-1.017975E-07
S.E. =	2.264843E-05

V12N1P14

C(0) =	1.000000207
C(1) =	.99999912
C(2) =	1.000000449
C(3) =	.99999992
C(4) =	1.000000005
C(5) =	.9999999999
Residual Variance =	8.723924E-12
Coeff of Det (R^2) =	1.000000
D(1) =	.0000002074
D(2) =	-2.98076E-07
D(3) =	-3.13736E-07
D(4) =	-.0000001455
D(5) =	.0000000014
D(6) =	.0000000049
D(7) =	-.0000001838
D(8) =	-.00000056
D(9) =	-.00000108
D(10) =	-.00000166
D(11) =	-.00000229
D(12) =	-.00000265
D(13) =	-.00000303
D(14) =	-.00000291
D(15) =	-.00000273
D(16) =	-.00000264
D(17) =	-.000002
D(18) =	-.000002
D(19) =	-.000003
D(20) =	-.000004
D(21) =	-.000007

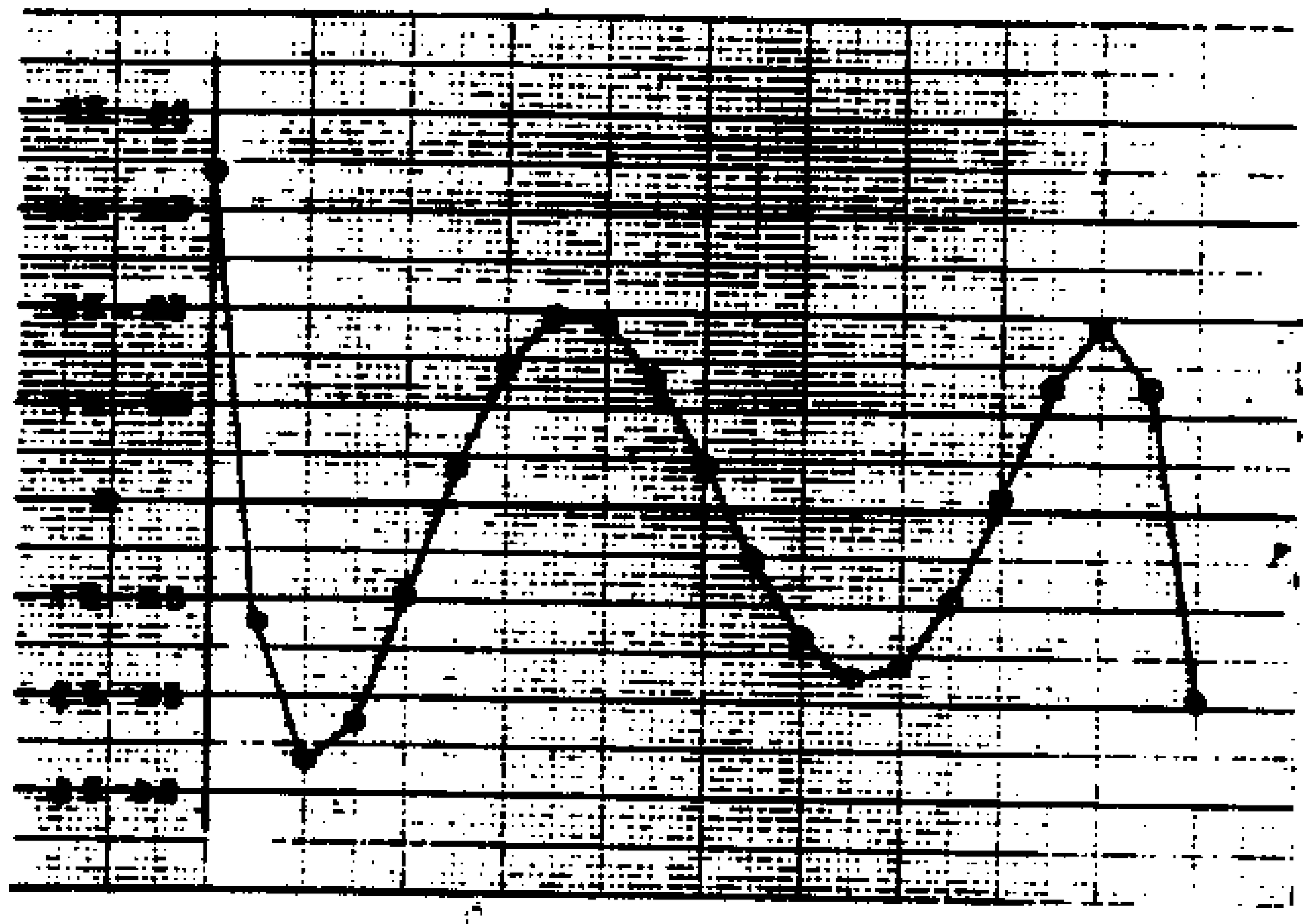
V12N1P24

The residuals for the V11N4P12 solution (the TI-74 using the Mathematica module matrix inversion routine) are all negative except for the first residual which is very small. Clearly, the mean of the residuals, which should be zero for a least squares polynomial curve fit, can not be zero. The residuals for the V12N1P24 solution (the TI-74 using Hood's orthogonal polynomial routine) show a similar negative bias, albeit much smaller in magnitude. For both programs the mean of the residuals was of the same order of magnitude as the standard error for the residuals. George Thomson had found a similar effect with a version of Hood's program on his PC where he observed a well defined ramp in the residuals even though the mean of the residuals was very small.

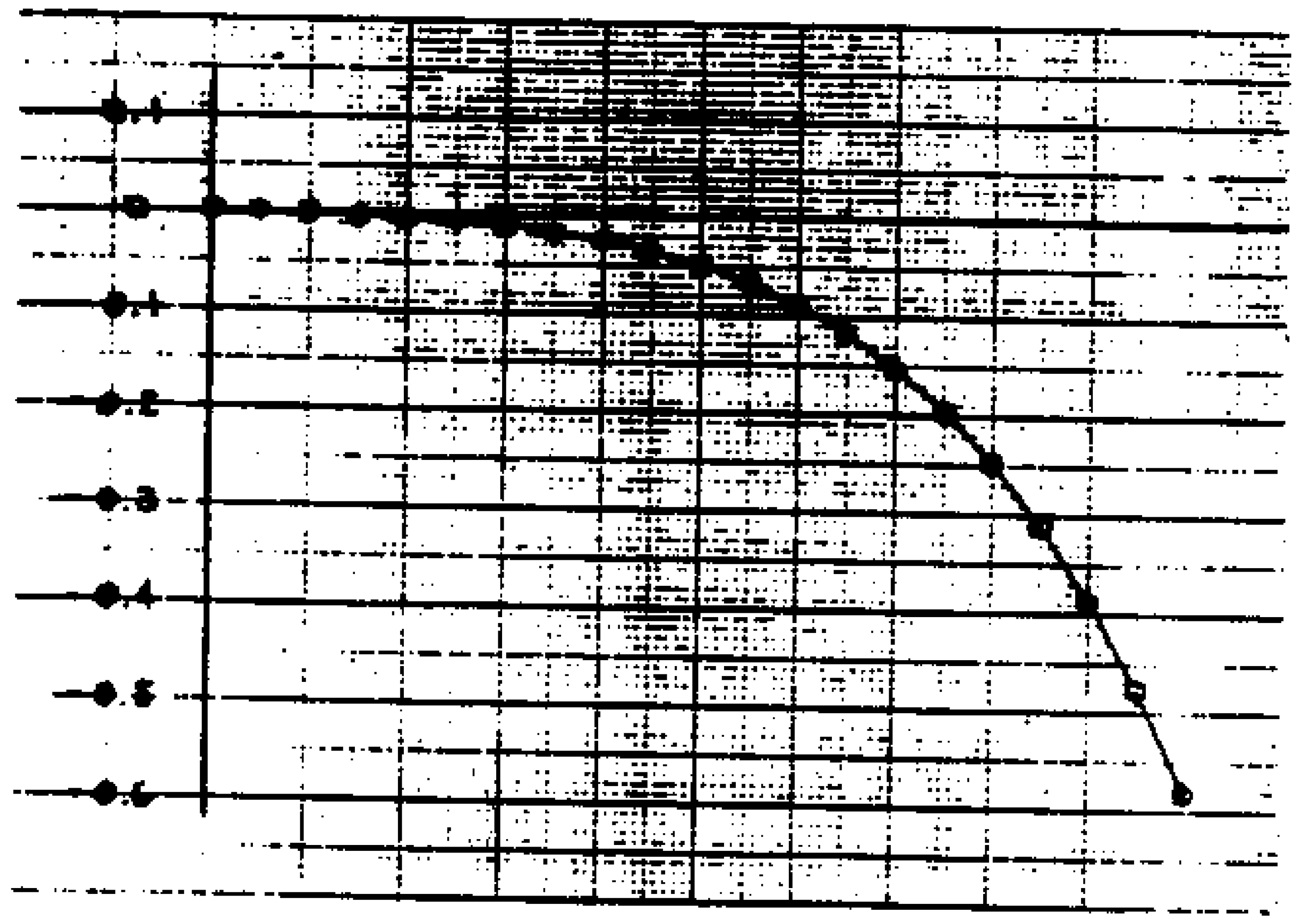
In contrast, the residuals for the V9N2P20 solution (the TI-59 using the ML-02 matrix inversion routine) and the V12N1P14 solution (the TI-74 using a row reduction routine) have both positive and negative signs such that the mean of the residuals is much smaller than the standard error for the residuals.

What is not evident from the tables is that the residuals for all four programs seem to be on smooth error curves. See the figures on the next page, but remember that the vertical scales vary considerably.

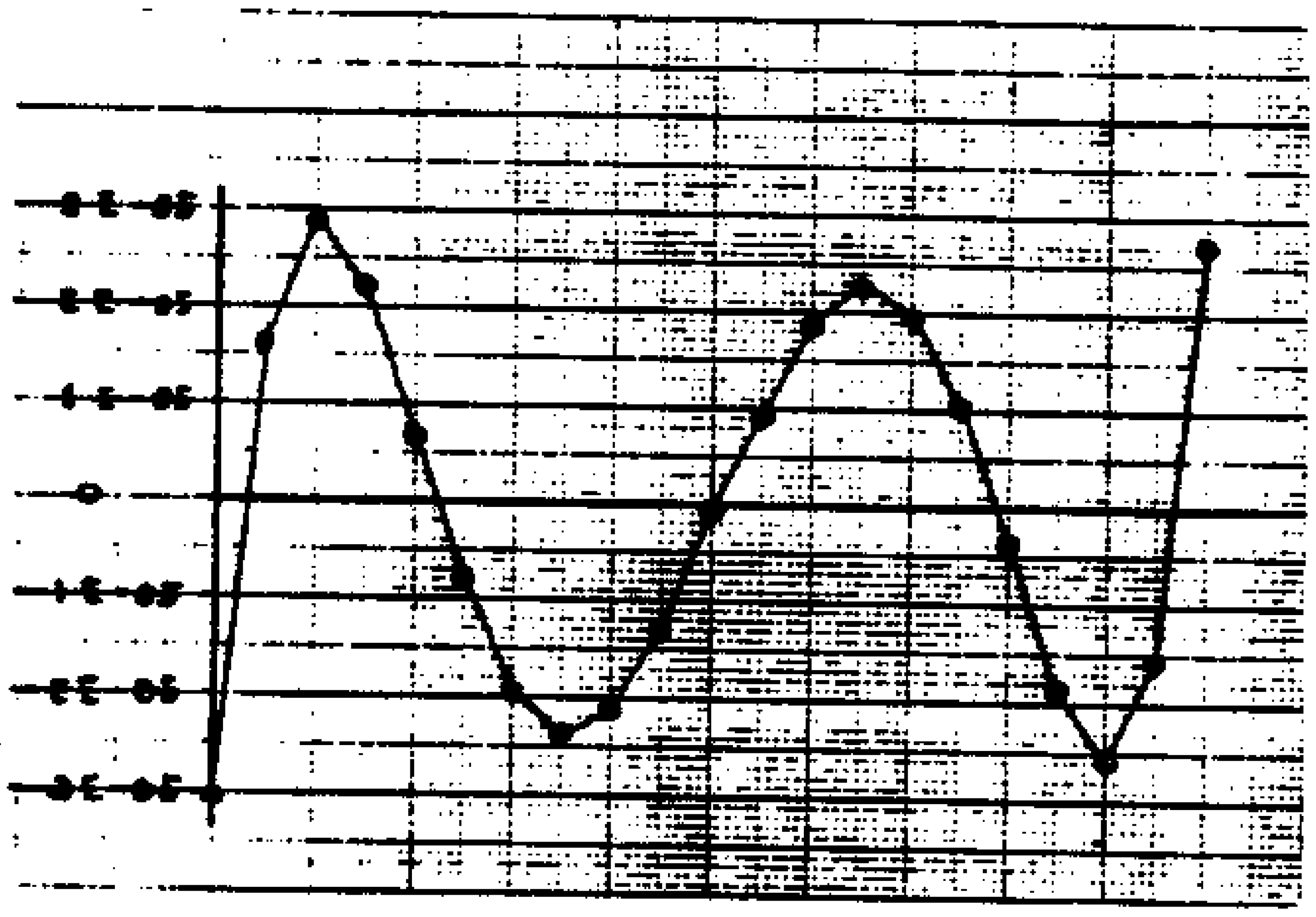
Evaluation of Polynomial Curve Fitting - (cont)



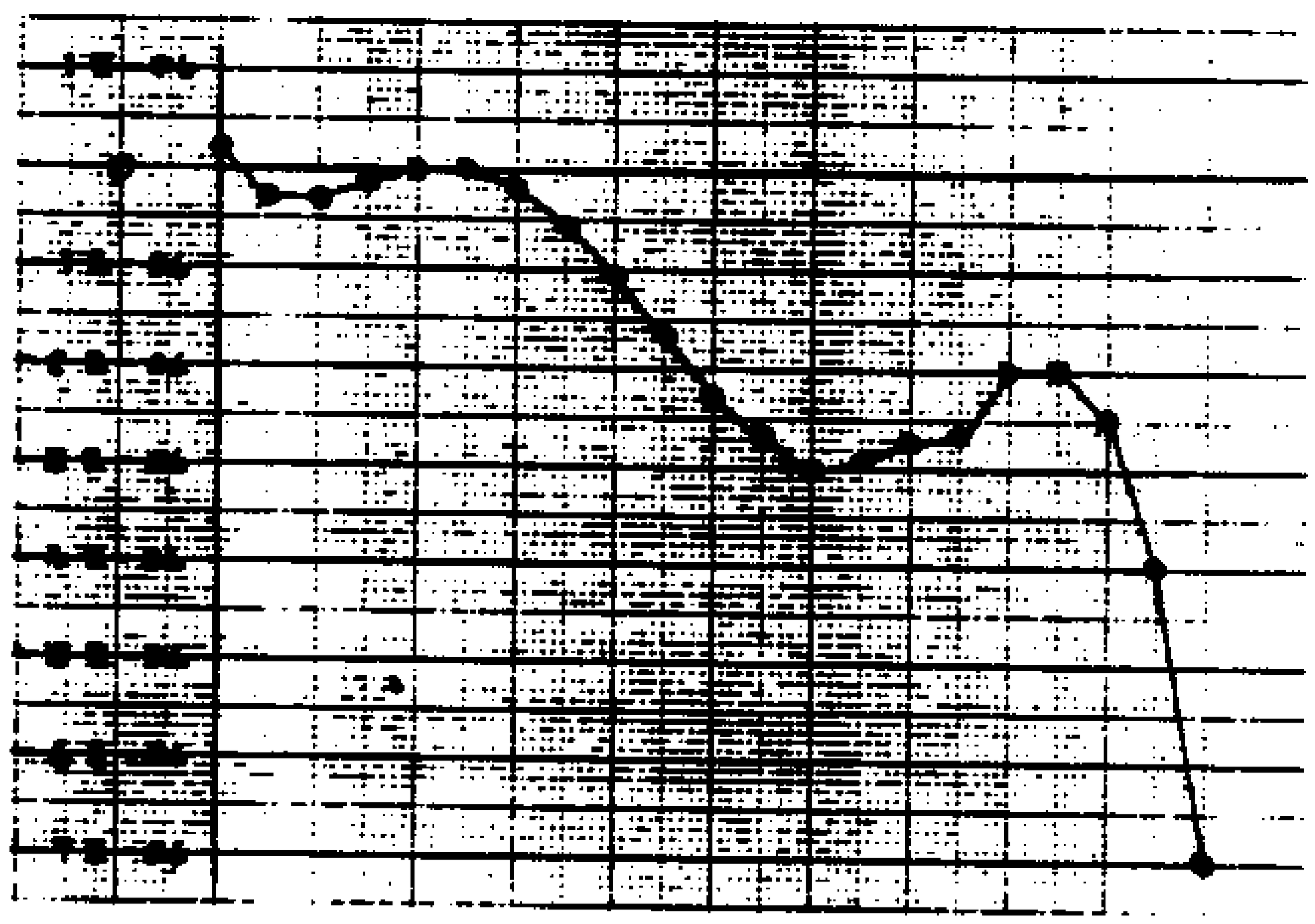
V9N2P20 - TI-59 Matrix Inversion



V11N4P12 - TI-74 Matrix Inversion



V12N1P14 - TI-74 Row Reduction



V12N1P24 - TI-74 Orthogonal Polynomial

The nature of these residual curves suggests that one might be able to find another set of polynomial coefficients which would fit the residual curves. Then, if the coefficients from the solution on the residuals were combined with the original coefficients, the resulting coefficients might provide a better fit to the original data. The table at the right shows the results after such an exercise with the polynomial regression program for the TI-59 (V9N2P20). The new coefficients have a C(ave) of 8.53 digits and a C(min) of 6.54 digits, a four digit improvement over the solution on page 17. The mean of the residuals is reduced by a factor of seven, and the standard error is reduced by a factor of over 1600. That kind of improvement in performance comes very hard with the TI-59 program where

1. The original set of 21 data pairs must be entered and each data pair takes about 20 seconds to be accepted.
2. The regression must be completed.
3. The original set of data pairs must be re-entered and the residuals calculated.
4. The set of data pairs for the residual errors must be entered. Again, each data pair requires about 20 seconds to be accepted.
5. The regression on the residuals must be completed, and the results combined with the original coefficients by hand.
6. The original data pairs must be re-entered and the new residuals calculated. This step can be easy this time if the user remembered to store the data entered in step 3 above.

A1	=	1.000000108	
A2	=	0.9999997117	
A3	=	1.000000121	
A4	=	0.9999999824	
A5	=	1.000000001	
A6	=	1.	
d1	=	1.08488	-07
d2	=	-7.6282	-08
d3	=	-1.1218	-07
d4	=	-7.04	-08
d5	=	0	
d6	=	6.6	-08
d7	=	1.02	-07
d8	=	1.1	-07
d9	=	1.	-07
d10	=	1.	-07
d11	=	-1.	-07
d12	=	2.	-07
d13	=	1.	-07
d15	=	5.	-07
d16	=	1.1	-06
d17	=	1.	-06
d18	=	1.	-06
d19	=	1.	-06
d20	=	1.	-06
d21	=	4.	-06
Mean	=	5.49-07	
S.E.	=	1.20-06	

Evaluation of Polynomial Curve Fitting - (cont)

Setting up an interative solution is much easier on the TI-74 where sufficient memory is available to hold the intermediate results. One question is how many iterations are needed. The following table helps answer that question for the solution which uses the matrix inversion routine from the Math module (V11N4P12).

A1 = .9989000000000E+00	A1 = .9999999833080E+00	A1 = .9999999830544E+00	A1 = .9999999549789E+00
A2 = .1002400000000E+01	A2 = .1000000061400E+01	A2 = .1000000052284E+01	A2 = .1000000127748E+01
A3 = .9996000000000E+00	A3 = .9999999685980E+00	A3 = .9999999762683E+00	A3 = .9999999445710E+00
A4 = .1000067000000E+01	A4 = .1000000005442E+01	A4 = .1000000003639E+01	A4 = .1000000008296E+01
A5 = .9999990000000E+00	A5 = .9999999996273E+00	A5 = .9999999997836E+00	A5 = .9999999995032E+00
A6 = .1000000100000E+01	A6 = .1000000000009E+01	A6 = .1000000000004E+01	A6 = .1000000000010E+01
d1 = 1.10000000000E-03	d1 = 1.66920000000E-08	d1 = 1.69455600000E-08	d1 = 4.50210600000E-09
d2 = -9.66100000000E-04	d2 = -1.83840000000E-08	d2 = -1.50330000000E-08	d2 = -3.51070000000E-08
d3 = -2.62320000000E-03	d3 = -1.83600000000E-08	d3 = -1.84730000000E-08	d3 = -4.74980000000E-08
d4 = -4.25230000000E-03	d4 = -3.80000000000E-09	d4 = -8.00000000000E-09	d4 = -2.55000000000E-08
d5 = -6.23440000000E-03	d5 = 1.14000000000E-08	d5 = 5.90000000000E-09	d5 = 6.90000000000E-09
d6 = -8.96250000000E-03	d6 = 1.93000000000E-08	d6 = 1.66000000000E-08	d6 = 3.42000000000E-08
d7 = -1.28536000000E-02	d7 = 1.64000000000E-08	d7 = 2.09000000000E-08	d7 = 4.82000000000E-08
d8 = -1.83607000000E-02	d8 = .00000000000E+00	d8 = 2.00000000000E-08	d8 = 4.00000000000E-08
d9 = -2.59848000000E-02	d9 = -1.00000000000E-08	d9 = 1.00000000000E-08	d9 = 3.00000000000E-08
d10 = -3.62869000000E-02	d10 = -4.00000000000E-08	d10 = .00000000000E+00	d10 = 1.00000000000E-08
d11 = -4.99000000000E-02	d11 = -7.00000000000E-08	d11 = -1.00000000000E-08	d11 = -2.00000000000E-08
d12 = -6.75411000000E-02	d12 = -9.00000000000E-08	d12 = .00000000000E+00	d12 = -4.00000000000E-08
d13 = -9.00232000000E-02	d13 = -1.10000000000E-07	d13 = 1.00000000000E-08	d13 = -3.00000000000E-08
d14 = -1.18267300000E-01	d14 = -1.20000000000E-07	d14 = 4.00000000000E-08	d14 = 1.00000000000E-08
d15 = -1.53314400000E-01	d15 = -1.40000000000E-07	d15 = 1.10000000000E-07	d15 = 7.00000000000E-08
d16 = -1.96337500000E-01	d16 = -1.70000000000E-07	d16 = 2.00000000000E-07	d16 = 1.60000000000E-07
d17 = -2.48654000000E-01	d17 = .00000000000E+00	d17 = 1.00000000000E-06	d17 = 1.00000000000E-06
d18 = -3.11736000000E-01	d18 = -1.00000000000E-06	d18 = .00000000000E+00	d18 = 1.00000000000E-06
d19 = -3.87225000000E-01	d19 = -1.00000000000E-06	d19 = .00000000000E+00	d19 = .00000000000E+00
d20 = -4.76942000000E-01	d20 = -1.00000000000E-06	d20 = 1.00000000000E-06	d20 = .00000000000E+00
d21 = -5.82900000000E-01	d21 = -2.00000000000E-06	d21 = 1.00000000000E-06	d21 = 1.00000000000E-06
Mean Error = -.1332507143	Mean Error = -2.727025E-07	Mean Error = 1.618495E-07	Mean Error = 1.550579E-07
S.E. = .2543009854	S.E. = 6.875774E-07	S.E. = 4.513731E-07	S.E. = 4.506149E-07
S2 = .9700346675	S2 = 7.09144E-12	S2 = 3.056066E-12	S2 = 3.045806E-12
C = 4.35837688	C = 8.538493149	C = 8.696504536	C = 8.313573852

The baseline program from V11N4P12 was modified to print out the solution and the residuals in exponential format, and to add solutions for the sum of the squares of the residuals (S2) and the average figure of merit (C). The left-hand column is the baseline solution, the same as in the second column on page 17. The next three columns reflect the result of additional iterative solutions on the residuals. The table shows that there is substantial improvement after the first iteration, but little improvement after additional iterations. The coefficients after the first iteration show an improvement of 4.2 digits and the standard error is reduced by a factor of 370,000.

Tests of the iterative technique with the solution based on the row reduction routine (V12N1P14) also showed substantial improvement after one iteration but little improvement from additional iterations. The coefficients after the first iteration show an improvement of 2.6 digits and the standard error is reduced by a factor of 60.

Programs which provide a single iteration for both the matrix inversion routine and the row reduction routine appear on pages 20 and 21. Of course execution time is increased. The matrix inversion program requires 135 seconds to solve the quintic. The row reduction program requires 110 seconds.

In a future issue we will examine the effect of iterative techniques on the orthogonal polynomial routine, and will address polynomial regression and iterative techniques for the TI-95.

<u>ITERATIVE REGRESSION USING THE TI-74 MATH MODULE</u> - This program is an iterative version of the program which appeared in V11N4P12/13. The least squares solution was moved to a subroutine (lines 900 to 995), an array for accumulation of the coefficients was added, and a working array, Z(50), to hold the dependent variables for use in the solution was added. Lines 130 and 140 provide for data input. Lines 150-170 select the order of the solution, clear the coefficient array, and transfer the input independent variables to the working array for use in the first pass least squares solution. The GOSUB 900 command at line 200 provides the first pass. The GOSUB 900 command at line 210 provides the second pass using the residuals from the first pass as the dependent variables. Additional iterations could be added by inserting more GOSUB 900 commands between lines 210 and 300. Lines 300-650 provide output of the solution to a printer or to the display. The PRINT USING commands at lines 340 and 550 define exponential format for maximum resolution in the output. Lines 700-740 provide options for different solutions without re-entry of the input data.

<u>User Instructions</u>

1. The Mathematics software module must be installed.

2. A full set of prompts are available.

3. The user defined functions must be defined by F(1) through F(N) in subroutine 800. To obtain a constant in the solution define one of the functions as one as in line 810 of the program below. Each pair of X,Z values are available in turn at entry to this subroutine. The user defined functions in the program listing below provides for a polynomial solution.

```
100 DIM A(8,8),B(8),C(8,
8),C1(8),F(8),X(50),Y(50
),Z(50)
110 CALL UP("Least Squar
es Fit",PN)
120 PRINT "Are the funct
ions correct?":PAUSE 2
130 INPUT "Number of Dat
a Pairs? ";K
140 CALL AU("X","Y",X(),
Y(),1,K,PN)
150 INPUT "Order of the
solution? ";N
160 FOR I=1 TO N:C1(I)=0
:NEXT I
170 FOR I=1 TO K:Z(I)=Y(
I):NEXT I
200 GOSUB 900
210 GOSUB 900
300 PRINT #PN
310 FOR I=1 TO N
320 X$="A"&STR$(I)&" = "
330 PRINT #PN,X$;
340 PRINT #PN,USING"##.#
##########^^^^",C1(I)
350 IF PN=0 THEN PAUSE
360 NEXT I
370 PRINT #PN
500 INPUT "Display Resid
uals (Y/N)? ";A$
510 IF A$="N"OR A$="n"TH
EN 600
520 FOR L=1 TO K
530 P$="d"&STR$(L)&" = "
540 PRINT #PN,P$;
550 PRINT #PN,USING"##.#
##########^^^^",Z(L)
560 IF PN=0 THEN PAUSE
570 NEXT L
580 PRINT #PN
600 PRINT #PN,"Mean Erro
r = ";S1/K
610 IF PN=0 THEN PAUSE
620 PRINT #PN
630 PRINT #PN,"S.E. = ";
SQR(S2/(K-N))
640 IF PN=0 THEN PAUSE
650 PRINT #PN
700 INPUT "Edit Input Da
ta (Y/N)? ";A$
710 IF A$="N"OR A$="n"TH
EN 730
720 CALL AU("X","Y",X(),
Y(),1,K,PN)
730 INPUT "Different ord
er (Y/N)? ";A$
740 IF A$="Y"OR A$="y"TH
EN 150
799 STOP
800 REM User Defined Fun
ctions
810 F(1)=1
820 FOR W=2 TO N
830 F(W)=F(W-1)*X(L)
840 NEXT W
890 RETURN
900 REM Normal Equation
Subroutine
905 FOR I=1 TO N:FOR J=1
 TO N
910 A(I,J)=0:NEXT J
915 B(I)=0:NEXT I
920 FOR L=1 TO K:GOSUB 8
00
925 FOR I=1 TO N:FOR J=1
 TO N
930 A(I,J)=A(I,J)+F(I)*F
(J):NEXT J
935 B(I)=B(I)+F(I)*Z(L):
NEXT I
940 NEXT L
945 CALL MATS(A(,),C(,),
B(),1,1,5,1,N,1,R)
950 IF R=-1 THEN PRINT "
Matrix is singular":PAUS
E
955 FOR I=1 TO N:C1(I)=C
1(I)+A(1,I):NEXT I
960 S1=0:S2=0
965 FOR L=1 TO K:GOSUB 8
00
970 YF=0:FOR J=1 TO N
975 YF=YF+C1(J)*F(J):NEX
T J
980 Z(L)=Y(L)-YF
985 S1=S1+Z(L):S2=S2+Z(L
)*Z(L)
990 NEXT L
995 RETURN
```

ITERATIVE REGRESSION USING A ROW REDUCTION ROUTINE - This program is an iterative version of the program which appeared in V12N1P14. In this case the least squares solution was moved to a subprogram SOLVE (lines 1000 to 1400) to provide another demonstration of the subprogram capability, including the calling of a subroutine within the subprogram. A subroutine mechanization such as that used on page 20 would actually yield a somewhat more efficient program. An array for accumulation of the coefficients was added, and two working arrays, R(50) and Z(50), were added.

Lines 105 to 190 provide for setup and data input. Lines 200-210 select the order of the solution, and clear the coefficient array. The CALL SOLVE command at line 300 provides the first pass. The CALL SOLVE command at line 310 provides the second pass using the residuals from the first pass as the dependent variables. Additional iterations could be added by inserting more CALL SOLVE commands between lines 310 and 400. Lines 400-650 provide output of the solution to a printer or to the display. The PRINT USING commands at lines 425 and 545 define exponential format for maximum resolution in the output. Lines 700-790 provide options for different solutions without re-entry of the input data. The user defined functions inside the subprogram at steps 1300-1350 are for a polynomial regression.

```
100 DIM A(8,8),B(8),C(8)
,F(8),X(50),Y(50),Z(50),
R(50)
105 INPUT "Use Printer <
Y/N>? ";A$
110 IF A$="Y"OR A$="y"TH
EN PN=1 ELSE 125
115 INPUT "Device Code ?
 ";P$
120 OPEN #1,P$,OUTPUT
125 PRINT "Are the funct
ions correct?":PAUSE 2
130 INPUT "Number of Dat
a Pairs? ";K
140 FOR I=1 TO K
150 A$="X"&STR$(I)&" = "
:INPUT A$:X(I)
160 IF PN<>0 THEN PRINT
#PN,A$,X(I)
170 A$="Y"&STR$(I)&" = "
:INPUT A$:Y(I)
180 IF PN<>0 THEN PRINT
#PN,A$,Y(I)
185 PRINT #PN:IF E$<>""T
HEN 700
190 NEXT I
200 INPUT "Order of the
solution? ";N
210 FOR I=1 TO N:C(I)=0:
NEXT I
300 CALL SOLVE(X(),Y(),Y
(),R(),C(),N,K,S1,S2)
310 CALL SOLVE(X(),Y(),R
(),R(),C(),N,K,S1,S2)
400 FOR I=1 TO N
410 X$="A"&STR$(I)&" = "
420 PRINT #PN,X$;
425 PRINT #PN,USING"##.#
##########^^^^",C(I)
430 IF PN=0 THEN PAUSE
440 NEXT I
450 PRINT #PN
500 INPUT "Display Resid
uals <Y/N>? ";A$
510 IF A$="N"OR A$="n"TH
EN 600
520 FOR L=1 TO K
530 P$="d"&STR$(L)&" = "
540 PRINT #PN,P$;
545 PRINT #PN,USING"##.#
##########^^^^",R(L)
550 IF PN=0 THEN PAUSE
560 NEXT L
570 PRINT #PN
600 PRINT #PN,"Mean = ";
S1/K
610 IF PN=0 THEN PAUSE
620 PRINT #PN
630 PRINT #PN,"S.E. = ";
SQR(S2/(K-N))
640 IF PN=0 THEN PAUSE
650 PRINT #PN
700 INPUT "Edit Input Da
ta <Y/N>? ";E$
710 IF E$="N"OR E$="n"TH
EN 780
720 INPUT "Which Data Pa
ir to Edit? ";I
730 IF I<1 OR I>K THEN 7
00
740 GOTO 150
780 INPUT "New Solution
<Y/N>? ";A$
790 IF A$="Y"OR A$="y"TH
EN 200
799 STOP
1000 SUB SOLVE(X(),Y(),Z
(),R(),C(),N,K,S1,S2)
1010 FOR I=1 TO N:FOR J=
1 TO N
1015 A(I,J)=0:NEXT J
1020 B(I)=0:NEXT I
1025 FOR L=1 TO K
1030 GOSUB 1300
1035 FOR I=1 TO N:FOR J=
1 TO N
1040 A(I,J)=A(I,J)+F(I)*
F(J):NEXT J
1045 B(I)=B(I)+F(I)*Z(L)
:NEXT I
1050 NEXT L
1100 FOR L=1 TO N
1105 P=A(L,L)
1110 FOR J=L TO N
1115 A(L,J)=A(L,J)/P:NEX
T J
1120 B(L)=B(L)/P
1125 FOR I=1 TO N
1130 IF I=L THEN 1155
1135 G=A(I,L)
1140 FOR J=L TO N
1145 A(I,J)=A(I,J)-G*A(L
,J):NEXT J
1150 B(I)=B(I)-G*B(L)
1155 NEXT I
1160 NEXT L
1200 FOR I=1 TO N:C(I)=C
(I)+B(I):NEXT I
1205 S1=0:S2=0
1210 FOR L=1 TO K:GOSUB
1300
1215 YF=0:FOR J=1 TO N
1220 YF=YF+C(J)*F(J):NEX
T J
1225 R(L)=Y(L)-YF
1230 S1=S1+R(L):S2=S2+R(
L)*R(L)
1235 NEXT L
1240 SUBEXIT
1300 REM USER DEFINED FU
NCTIONS
1310 F(1)=1
1320 FOR W=2 TO N
1330 F(W)=F(W-1)*X(L)
1340 NEXT W
1350 RETURN
1400 SUBEND
```

CUBIC SOLUTION - Larry Leeds. V11N4P16/17 examined the results for a variety of cubic solutions, and invited members to submit more accurate solutions. The algorithm for the program presented here is as follows: Zero is used as the first approximation. An Exact Newton (the program does the differentiation) finds one of the real roots, or the only one. This value is truncated to a six digit number which is used as the approximation. Newton then produces an exceptionally close answer and prints the root. Synthetic division derives the residual quadratic. A test is made to determine if the roots are complex or real, and the quadratic is solved. If complex, the roots are printed. If real, the two roots are truncated to 6 digit numbers which are considered as approximations. Each root, in turn, is then presented to the original cubic and solved by the exact Newton.

```
Ax^3 + Bx^2 + Cx + D = 0
A =   1
B =  -2
C =   1.333333333
D =  -.2222222222

Real Root =
 2.466929833686E-001
Complex Roots =
Real Part
 8.766535083155E-001
Imaginary Part
+/- 3.637078786574E-001

Ax^3 + Bx^2 + Cx + D = 0
A =   1
B =  -6.975
C =   16.216874
D =  -12.5680758

Real Root =
 2.324000000000E+000
Real Root =
 2.326000000000E+000
Real Root =
 2.325000000000E+000
```

The improvement obtained from truncation is presumed to be associated with the Newton iteration process and the overflow truncation which occurs in the computer. Each iteration produces a 14 digit number which will be both squared and cubed, hence the value of the function is not exact. As the iteration approaches the correct value of X you may obtain a value which is in error in both the 13th and 14th digit positions. Because of overflow truncation, the program would conclude that the erroneous value of X is the exact root. I found by experiment that if this value of X is then truncated and is put through the Newton solution again, the result will be very close to the exact value. The examples at the right show the effect of the truncation. The solutions at the top are with the program as on page 11. The solutions at the bottom are with the truncations removed. Note that the program finds the exact roots for the second example when the truncation is in place.

```
Ax^3 + Bx^2 + Cx + D = 0
A =   1
B =  -2
C =   1.333333333
D =  -.2222222222

Real Root =
 2.466929833685E-001
Complex Roots =
Real Part
 8.766535083160E-001
Imaginary Part
+/- 3.637078786559E-001

Ax^3 + Bx^2 + Cx + D = 0
A =   1
B =  -6.975
C =   16.216874
D =  -12.5680758

Real Root =
 2.324000817341E+000
Real Root =
 2.325999823508E+000
Real Root =
 2.324998366829E+000
```

I would appreciate hearing from the club members of any difficulties encountered with this algorithm.

Editor's Note: Larry writes his BASIC programs in Microsoft BASIC for the Radio Shack Model 100. I convert them for use with the CC-40 and TI-74. The truncation function is very easy to implement on the Model 100 by using the single precision command, e.g., A = CSNG(A) truncates the precision from 14 digits to 7 digits. The subprogram in lines 1001-1004 solves the problem nicely for the TI-74 or CC-40. The calls at lines 240, 250 and 800 pass the appropriate value to and from the subprogram, and the subprogram uses its own variables S and K. Note that it is permissible to use K for one purpose inside the subprogram even though it is defined differently outside the subprogram and in the call to the subprogram. One alternative was to repeat lines 1002 and 1003 three times with appropriately chosen variables. Another was to call a subroutine -- that would require a change of variables prior to entry to the subroutine and after exit from the subroutine.

Line 100 of this program illustrates the use of the PAUSE ALL command to stop the computer for display if a printer is not used.

Cubic Solution - (cont)

```
10 REM Cubic Solution by
 L. Leeds
12 INPUT "Use Printer (Y
/N) ? ";A$
14 IF A$="Y"OR A$="y"THE
N PN=1 ELSE 20
16 INPUT "Device Code ?
";P$
18 OPEN #1,P$,OUTPUT
20 PRINT #PN:PRINT #PN,"
Ax^3 + Bx^2 + Cx + D = 0
"
25 IF PN=0 THEN PAUSE
30 IMAGE "##.###########
#^^^^^^";N
40 R$="Real Root = "
60 A$="A =  ":INPUT A$:A
65 PRINT #PN,A$;A
70 B$="B =  ":INPUT B$:B
75 PRINT #PN,B$;B
80 C$="C =  ":INPUT C$:C
85 PRINT #PN,C$;C
90 D$="D =  ":INPUT D$:D
95 PRINT #PN,D$;D
100 PRINT #PN:PAUSE ALL
110 P=B/A:Q=C/A:R=D/A
120 A=P:B=Q:C=R
130 F=1:X=0:E=1.E-08
140 GOSUB 600
150 Y=X-U/V
160 IF E>ABS(X-Y)THEN 18
0
170 X=Y:GOTO 140
180 IF F=1 THEN 800
190 PRINT #PN,R$:PRINT #
PN,USING 30,Y
200 G=A+Y:H=G*Y+B
210 M=G*G/4:J=M-H:G=-G/2
220 IF J<0 THEN J=-J:Z=S
QR(J):GOTO 700
230 Z=SQR(J):K=G+Z:N=G-Z
240 CALL TRUNC(K)
250 CALL TRUNC(N)
260 X=K
270 GOSUB 600
280 Y=X-U/V
290 IF E>ABS(X-Y)THEN P=
Y:PRINT #PN,R$:PRINT #PN
,USING 30:P:GOTO 310
300 X=Y:GOTO 270
310 X=N
320 GOSUB 600
330 Y=X-U/V
340 IF E>ABS(X-Y)THEN Q=
Y:PRINT #PN,R$:PRINT #PN
,USING 30:Q:GOTO 900
350 X=Y:GOTO 320
600 U=X*X*(X+A)+B*X+C
610 V=X*(3*X+2*A)+B
620 RETURN
700 IF ABS(G)<9.E-13 THE
N G=0
710 PRINT #PN,"Complex R
oots = "
720 PRINT #PN,"Real Part
":PRINT #PN,USING 30:G
730 PRINT #PN,"Imaginary
 Part"
740 PRINT #PN,"+/-";
750 PRINT #PN,USING 30,Z
760 GOTO 900
800 CALL TRUNC(Y)
820 F=0:X=Y:GOTO 140
900 INPUT "Another Probl
em (Y/N) ? ";A$
910 IF A$="Y"OR A$="y"TH
EN PAUSE 0:GOTO 20
920 END
1000 REM Subprograms
1001 SUB TRUNC(S)
1002 K=-INT(LOG(ABS(S)))
+5
1003 S=INT(S*10^K)*10^(-
K)
1004 SUBEND
```

MORE PERIPHERALS FOR THE TI-74? - P. Hanson. In early spring I was contacted by a telephone survey performed for TI. Purchasers of the TI-74 and TI-95 were being surveyed. The questions suggest that several new peripherals are being considered for those devices:

- An interface which would allow the connection of the AC-9201 to the TI-74 without the use of the PC-324.
- A Centronics interface for use with printers and plotters.
- An RS-232 interface.
- A combined Centronics and RS-232 interface.

One of the purposes of the survey was to determine what might be acceptable prices. The caller asked what I thought might be a price at which I would probably buy, and then asked the liklihood that I would buy at various other price ranges. Nothing in the catalogs yet. We will have to wait and see.

BOOK REVIEWS:

STATISTICS LIBRARY. Application Software for the Sharp EL-5500 and PC-1403 Scientific Computers. Maurice E. T. Swinnen and David Thomas. Systems Publications, Box 300488, Arlington, TX 76010. 1987. 105 pages. Paperback. $11.95.

The book contains 23 BASIC language programs for the solution of typical problems in statistics. The programs are:

- Cauchy Distribution Curve Fit
- Circle Best Fit
- Chi-Square Distribution
- Contingency Table
- #*Exponential Curve Fitting
- Gaussian Distribution
- Histogram
- *Hyperbolic Curve Fitting
- #*Logarithmic Curve Fitting
- Mann-Whitney Ranked-Sum Test
- Means and Moments
- Multiple Linear Regression with 2 Variables
- Multiple Linear Regression with 3 Variables
- One-Way ANOVA
- #*Parabolic Curve Fitting
- Poisson Distribution
- #*Power Curve Fitting
- *Reciprocal Curve Fitting
- #*Straight Line Curve Fitting
- Student's t-Distribution
- T-Test for Paired Observations
- T-Test for Unpaired Observations
- Two-Way ANOVA

The documentation for each program includes program listings, step-by-step user instructions, and sample problems. Although the programs were written for the Sharp machines they are easily convertible for use on the TI-74. An example conversion appears on pages 4 and 5 of this issue.

I have started working my way through this book, including a cross check of the results against those from other published programs. I have verified the results within the limits of differences between machines for the programs identified with a # sign in the table above. There are some minor errors in the book:

a. Page 5 states that the Cauchy distribution has "... the rather peculiar characteristic of a mean but no standard deviation". References such as the Mathematical Dictionary by James and James, and Mathematical Methods by Cramer, indicate that the Cauchy distribution may have a mode and a median, but not a mean or a standard deviation since no moments of positive order are finite. William Volk made a similarly incorrect statement on page 76 of the second edition of his book Curve Fitting for Programmable Calculators which was reviewed in V8N2P20.

b. The equation for the logarithmic curve on page 35 should be y = a + b(LN(x)). The equation is correct in the program listings on page 37.

c. The programs which require multiple input such as the curve fitting programs accept the input as string values and convert the strings to numerics internally with VAL commands. This permits termination of the input by entering the letter E. A prompt "End Input by entering E" is provided to remind the user of this feature; however, for seven of the programs the address for an earlier GOTO is improper such that the prompt will not be seen. The programs with that discrepancy have an asterisk in the listing above. In each case the problem can be corrected by changing line 40 to IF PF=0 GOTO 60.

Old-timers will remember Maurice Swinnen as the editor of TI PPC Notes from 1980 through 1982. Maurice has arranged a twenty per cent discount to club members on this book and the two books which follow. Order from the address listed above, and mention TI PPC Notes when you order. There is a $2.00 dollar shipping and handling charge for an order of one book, or a $3.00 charge for two or more books.

Book Reviews - (cont)

ELECTRICAL ENGINEERING LIBRARY. Application Software for the Sharp EL-5500 and PC-1403 Scientific Computers. Maurice E. T. Swinnen and David Thomas. Systems Publications, Box 300488, Arlington, TX 76010. 1987. 119 pages. Paperback. $11.95.

This second book in the series of libraries of BASIC programs is available now. The included programs are:

- Active Band-Pass Filter
- Active High-Pass Filter
- Active Low-Pass Filter
- Biomedical Filtering Circuit
- Bit-Error Probability
- Bode/Nyquist Calculation
- Gate-Bias FET Circuit
- Logarithmic Conversions
- Low Frequency Transistor Amplifier
- Odd Resistance Value Synthesizer
- Passive Filter Design
- Phase Locked Loop Design
- Power Supply Filter Design
- Rating Unknown Power Transformers
- Reactance Chart
- Self-Bias FET Circuit
- Serial-to-Parallel Conversion
- Single-Layer Coil Design
- T- and PI-Pad Attenuators
- Temperature Conversions
- Transistor Parameter Conversions
- Twin-T Circuit Design
- Voltage Feedback FET Circuit
- Zener Diode Power Supply

MATHEMATICS LIBRARY. Application Software for the Sharp EL-5500 and PC-1403 Scientific Computers. Maurice E. T. Swinnen and David Thomas. Systems Publications, Box 300488, Arlington, TX 76010. 1988. 120 pages. Paperback. $11.95.

This third book in the series of libraries of BASIC programs will become available in early 1988. The included programs will cover subjects such as

Complex Functions: add, subtract, multiply, divide
Complex Functions: square, square root, reciprocal, log, exponent, polar to rectangular, and rectangular to polar.
Complex Trigonometric Functions: sine, cosine, tangent, arcsin, arccos and arctan.
Complex Functions: Y^X, Y^(1/X) and log to the base x of y.

- Differential Equations, Runge-Kutta
- Gamma Function
- Gauss Quadrature INtegration
- 3D Coordinate Transformations
- Complex Roots of a Complex Number
- Derivatives
- Multiprecision Division
- Polynomial Function Evaluation
- Extended Precison Factorials
- Fourier Series
- Three Point Interpolation
- Polynomial Addition and Subtraction
- Polynomial Multiplication
- Polar to Rectangular
- Rectangular to Polar
- Quadratic Equations
- Coordinate Translation
- Rational Fractions
- Simultaneous Equations

ERROR IN PROGRAMMABLE CALCULATOR NEWS - A copy of the Volume 1 Number 1 issue was included with V12N1. A TI representative writes that there were errors in the listing of Patrick Hicks' program "Can I Really Afford It" on page 8 of the newsletter. Lines 190 and 230 should have read as follows:

```
190  PV=PMT*(1-(1+I)^(-N))/I

230  PMT=PV*I/(1-(1+I)^(-N))
```

<u>CONVERSION OF SHARP BASIC TO TI-74 AND CC-40 BASIC</u> - P. Henson. The books reviewed on pages 4 and 5 include program listings in BASIC; however; the listings contain commands which are unique to Sharp BASIC. Maurice Swinnen has given me permission to reproduce one of the programs from the <u>Statistics Library</u> to illustrate the conversion process. The program selected was that for Power Curve Fitting from pages 73-75. The listing in the left-hand column on the facing page is from page 75 of the book. The listing in the center column is a conversion for the TI-74 and CC-40. The mathematics portions of the program can be used as is by the TI-74 or CC-40. Most of the input/output routines require some change. You should try to understand each change in order to be able to translate other programs. Representative changes are:

In line 40 the GOTO has been changed to THEN to accommodate the differences between the two BASICs. A similar change is required at lines 70, 100, 110, 260, 270, and 290. The THEN address is changed to 60 to permit access to the prompt on ending input of data.

The BEEP 2 commands in lines 60, 70, 260, 280, 290 and 380 are deleted since the TI-74 does not have that capability. CC-40 users should replace BEEP 2 with DISPLAY BEEP.

The PAUSE ALL command inserted at line 210 stops the program when any subsequent sequence completes a line in the display.

The IMAGE command at line 230 provides format control for the PRINT USING commands in lines 240 and 250.

Lines 320-360 are changed to convert the "Use Printer" selection to TI BASIC.

The listing in the right-hand column on page 7 is an enhancement to provide output of residual errors. The changes relative to the program in the center column are:

Line 15 provides the dimension statement to set up the arrays to hold the input data pairs for use in the residual calculations. If you are going to use more than twenty data pairs you should change to dimensions accordingly.

The statement N=0 is added to line 20 to zero the counter of data pairs.

Line 145 increments the counter and stores the input data pairs in the arrays.

Line 255 permits the user to elect output of residuals.

Lines 500-580 are the subroutine which calculates the residuals, and the mean and RMS of the residuals.

Parentheses which were not needed have been deleted in lines 160 through 200.

Prinntouts for the sample problem from the book appears at the right. Note that the mean of the residuals is not very close to zero. This occurs because of the transformation used to linearize the power equation for the least squares calculation. The least squares calculation is in the linearized variables, LN(X) and LN(Y), not in the variables X and Y. The mean of the residuals from the equation LN(Y) = LN(a) + b*LN(X) is -1.8E-14, near zero as expected.

```
Power  y = ax^b

X=  1
Y=  3.2
X=  2
Y=  7.4
X=  3
Y=  12
X=  4
Y=  16.8
X=  5
Y=  22
a=  3.211745293
b=  1.196427406
RR=            .99996805
Corr.RR=       .99995741
X=  3
Y=  11.9558936
```

```
Power  y = ax^b,

X=  1
Y=  3.2
X=  2
Y=  7.4
X=  3
Y=  12
X=  4
Y=  16.8
X=  5
Y=  22
a=  3.211745293
b=  1.196427406
RR=            .99996805
Corr.RR=       .99995741
d1=-.011745293
d2= .0395963302
d3= .044106398
d4=-.067944759
d5=-.0296686814
d ave = -.0051312011
d RMS =  .0427736287
```

Conversion of Sharp BASIC to TI BASIC - (cont)

```
10:"C":WAIT 150:PRINT "
      Power  y=ax^b"
20:T1=0:T8=0:T9=0:U0=0:
   U1=0:U2=0
30:GOSUB 310:USING
40:IF PF=0 GOTO 70
50:PRINT "    Power  y=
   ax^b"
60:BEEP 2:INPUT "End in
   put by entering E",Z
   Z$
70:BEEP 2:INPUT "X=? ";
   XX$:IF XX$="E" GOTO
   160
80:X=VAL (XX$):IF PF=0
   GOTO 100
90:PRINT "X= ";X
100:BEEP 2:INPUT "Y=? ";
   YY$:IF YY$="E" GOTO
   160
110:Y=VAL (YY$):IF PF=0
   GOTO 130
120:PRINT "Y= ";Y
130:T1=T1+1:T8=T8+LN (X)
   :T9=T9+LN (X)^2:U0=U
   0+LN (Y):U1=U1+LN (Y
   )^2
140:U2=U2+(LN (X)*LN (Y)
   )
150:GOTO 70
160:R5=(T1*T9)-(T8*T8)
170:S1=((T9*U0)-(T8*U2))
   /R5:A=EXP (S1)
180:S2=((T1*U2)-(T8*U0))
   /R5:B=S2
190:RR=((S1*U0)+(S2*U2)-
   ((U0*U0)/T1))/(U1-((
   U0*U0)/T1))
200:RS=1-((1-RR)*(T1-1)/
   (T1-2))
210:PRINT "a= ";A
220:PRINT "b= ";B
230:USING "#####.#######
   #"
240:PRINT "RR=      ";RR
250:PRINT "Corr.RR=";RS:
   USING
260:BEEP 2:INPUT "Predic
   t Y? Y/N ";Z$:IF Z$=
   "N" OR Z$="n" GOTO 2
   80
270:GOSUB 370:BEEP 2:
   INPUT "Y again? Y/N
   ";Z$:IF Z$="Y" OR Z$
   ="y" GOTO 270
280:BEEP 2:INPUT "Add mo
   re data? Y/N ";Z$:IF
   Z$="Y" OR Z$="y"
   GOTO 70
290:BEEP 2:INPUT "New ca
   lculation? Y/N ";Z$:
   IF Z$="Y" OR Z$="y"
   GOTO 20
300:END
310:REM  ***Printer?***
320:PF=0:WAIT :BEEP 2:
   INPUT "Printer? Y/N
   ";N$
330:IF N$="N" OR N$="n"
   GOTO 350
340:PF=1:PRINT = LPRINT
   :PRINT "------------
   -------------":GOTO 3
   60
350:PRINT = PRINT
360:RETURN
370:REM ** Predict Y **
380:BEEP 2:INPUT "X=? ";
   XX:IF PF=0 GOTO 400
390:PRINT "X= ";XX
400:YY=A*XX^B
410:PRINT "Y= ";YY
420:RETURN
```

```
10 A$="Power  y = ax^b":
PRINT A$:PAUSE 2
20 T1=0:T8=0:T9=0:U0=0:U
1=0:U2=0
30 GOSUB 310
40 IF PF=0 THEN 60
50 PRINT #1,A$:PRINT #1
60 INPUT "End input by e
ntering E ":ZZ$
70 INPUT "X = ":XX$:IF X
X$="E"OR XX$="e"THEN 160
80 X=VAL(XX$):IF PF=0 TH
EN 100
90 PRINT #1,"X= ":X
100 INPUT "Y=? ":YY$:IF
YY$="E"OR YY$="e"THEN 16
0
110 Y=VAL(YY$):IF PF=0 T
HEN 130
120 PRINT #1,"Y= ":Y
130 T1=T1+1:T8=T8+LN(X):
T9=T9+LN(X)^2:U0=U0+LN(Y
):U1=U1+LN(Y)^2
140 U2=U2+(LN(X)*LN(Y))
150 GOTO 70
160 R5=(T1*T9)-(T8*T8)
170 S1=((T9*U0)-(T8*U2))
/R5:A=EXP(S1)
180 S2=((T1*U2)-(T8*U0))
/R5:B=S2
190 RR=((S1*U0)+(S2*U2)-
((U0*U0)/T1))/(U1-((U0*U
0)/T1))
200 RS=1-((1-RR)*(T1-1)/
(T1-2))
210 PAUSE ALL:PRINT #PF,
"a= ":A
220 PRINT #PF,"b= ":B
230 IMAGE #####.########
240 PRINT #PF,"RR=      "
;:PRINT #PF,USING 230,RR
250 PRINT #PF,"Corr.RR="
;:PRINT #PF,USING 230,RS
260 INPUT "Predict Y? Y/
N ":Z$:IF Z$="N"OR Z$="n
"THEN 280
270 GOSUB 370:INPUT "Y a
gain? Y/N ":Z$:IF Z$="Y"
OR Z$="y"THEN 270
280 INPUT "Add more data
? Y/N ":Z$:IF Z$="Y"OR Z
$="y"THEN 70
290 INPUT "New calculati
on? Y/N ":Z$:IF Z$="Y"OR
 Z$="y"THEN 20
300 END
310 REM ***Printer?***
320 IF PF=1 THEN CLOSE #
1:PF=0
330 INPUT "Use Printer?
Y/N ":N$
340 IF N$="Y"OR N$="y"TH
EN PF=1 ELSE 360
350 OPEN #1,"12",OUTPUT
360 PRINT #PF:RETURN
370 REM ** Predict Y **
380 INPUT "X= ":XX
390 PRINT #PF,"X= ":XX
400 YY=A*XX^B
410 PRINT #PF,"Y= ":YY
420 RETURN
```

```
10 A$="Power  y = ax^b":
PRINT A$:PAUSE 2
15 DIM U(20),V(20)
20 T1=0:T8=0:T9=0:U0=0:U
1=0:U2=0:N=0
30 GOSUB 310
40 IF PF=0 THEN 60
50 PRINT #1,A$:PRINT #1
60 INPUT "End input by e
ntering E ":ZZ$
70 INPUT "X = ":XX$:IF X
X$="E"OR XX$="e"THEN 160
80 X=VAL(XX$):IF PF=0 TH
EN 100
90 PRINT #1,"X= ":X
100 INPUT "Y=? ":YY$:IF
YY$="E"OR YY$="e"THEN 16
0
110 Y=VAL(YY$):IF PF=0 T
HEN 130
120 PRINT #1,"Y= ":Y
130 T1=T1+1:T8=T8+LN(X):
T9=T9+LN(X)^2:U0=U0+LN(Y
):U1=U1+LN(Y)^2
140 U2=U2+LN(X)*LN(Y)
145 N=N+1:U(N)=X:V(N)=Y
150 GOTO 70
160 R5=T1*T9-T8*T8
170 S1=(T9*U0-T8*U2)/R5:
A=EXP(S1)
180 S2=(T1*U2-T8*U0)/R5:
B=S2
190 RR=(S1*U0+S2*U2-U0*U
0/T1)/(U1-U0*U0/T1)
200 RS=1-(1-RR)*(T1-1)/(
T1-2)
210 PAUSE ALL:PRINT #PF,
"a= ":A
220 PRINT #PF,"b= ":B
230 IMAGE #####.########
240 PRINT #PF,"RR=      "
;:PRINT #PF,USING 230,RR
250 PRINT #PF,"Corr.RR="
;:PRINT #PF,USING 230,RS
255 INPUT "Display resid
uals? Y/N ":Z$:IF Z$="Y"
OR Z$="y"THEN GOSUB 500
260 INPUT "Predict Y? Y/
N ":Z$:IF Z$="N"OR Z$="n
"THEN 280
270 GOSUB 370:INPUT "Y a
gain? Y/N ":Z$:IF Z$="Y"
OR Z$="y"THEN 270
280 INPUT "Add more data
? Y/N ":Z$:IF Z$="Y"OR Z
$="y"THEN 70
290 INPUT "New calculati
on? Y/N ":Z$:IF Z$="Y"OR
 Z$="y"THEN 20
300 END
310 REM ***Printer?***
320 IF PF=1 THEN CLOSE #
1:PF=0
330 INPUT "Use Printer?
Y/N ":N$
340 IF N$="Y"OR N$="y"TH
EN PF=1 ELSE 360
350 OPEN #1,"12",OUTPUT
360 PRINT #PF:RETURN
370 REM ** Predict Y **
380 INPUT "X= ":XX
390 PRINT #PF,"X= ":XX
400 YY=A*XX^B
410 PRINT #PF,"Y= ":YY
420 RETURN
500 REM ** Residuals **
510 S1=0:S2=0
520 FOR I=1 TO N
530 D=V(I)-A*U(I)^B:S1=S
1+D:S2=S2+D*D
540 PRINT #PF,"d"&STR$(I
)&"=":D
550 NEXT I
560 PRINT #PF,"d ave = "
:S1/N
570 PRINT #PF,"d RMS = "
:SQR(S2/N)
580 RETURN
```

<u>ANOTHER SIMULTANEOUS EQUATION SOLUTION</u> - In V12N3P4-7 I reviewed three books of BASIC programs which were co-authored by former TI PPC Notes editor Maurice Swinnen. The programs are written in Sharp BASIC which contains some unique commands which are not available with the TI-74. An example conversion of a program from the <u>Statistics Library</u> was provided.

Although the contents of the <u>Mathematics Library</u> book were listed on V12N3P5, the book did not become available until after V12N3 was printed. When it arrived I converted the simultaneous equations program for use with the TI-74 so I could compare its capability with similar programs published in earlier issues. The listing for the converted program on page 13 was made using the HX-1000 and a cable like that described in V12N3P13. Comments on the conversion follow:

Lines 10 through 210 set the dimensioning, call the subroutine which selects the printer options, and provide for data input. Appropriate changes have been made to accommodate the differences between machines.

All of the programs in the <u>Mathematics Library</u> use the statement GOSUB "PRINTER?" in line 20 to call the printer option subroutine, and the first line of the subroutine is the statement "PRINTER?". This indicates that Sharp BASIC has a label capability that is not available on the TI-74 or CC-40. The GOSUB 800 statement at line 30 of the conversion provides the equivalent result. The programs in the <u>Statistics Library</u> and the <u>Electrical Engineering Library</u> did not use the label capability.

Lines 220 through 720 which mechanize the solution equations are nearly identical to the program in the book. The only changes are the replacement of GOTO with THEN in lines 230, 240, 350, 390, 420, 590, 710 and 720.

Lines 730 through 790 provide for output of the solution. Again, appropriate changes were made to accommodate the differences between machines.

Line 800 through 890 provide for selection of the printer option. The subroutine provides prompting for use of either the PC-324 or HX-1000. Line 880 selects the compressed print (36 characters per line) option of the HX-1000 to avoid the wraparound which would occur with the 18 character per line normal mode.

As with the conversion in V12N3P6-7 the BEEP 2 statements which appear in the book were deleted since the TI-74 does not have a BEEP capability. CC-40 users can replace the BEEP 2 statements with DISPLAY BEEP.

Lines 790 through 810 and 910 through 930 in the program in the book provide an option to solve another problem without going through the printer selection process again. That capability was not provided in the translation. To solve another problem with the translation simply run the program again. The RUN command zeroes all the variables.

A full set of prompts are provided with the program. Each equation is entered in order. The matrix of coefficients of the variables is stored in the two dimensional A array. The vector of constants is stored in the one dimensional B array. The solutions are derived in the X array.

Another Simultaneous Equation Solution - (cont)

Program Listing:

```
10 A$="Simultaneou
s Equations":PRINT
 A$:PAUSE 2
20 DIM A(22,22),B(
22),X(22)
30 GOSUB 800
40 IF PF=0 THEN 70
50 PRINT #1,A$
60 PRINT #1
70 INPUT "Number o
f equations = ? ";
N
100 FOR I=1 TO N
110 FOR J=1 TO N
120 II$=STR$(I):JJ
$=STR$(J):AA$="A("
&II$&","&JJ$&")= "
140 INPUT AA$;A(I,
J)
150 PRINT #PF,AA$;
A(I,J)
160 NEXT J
170 BB$="B("&II$&"
)= "
190 INPUT BB$;B(I)
200 PRINT #PF,BB$;
B(I)
210 PRINT #PF:NEXT
 I
220 Z=0
230 IF N<>0 THEN 2
90
240 IF A(1,1)=0 TH
EN 270
250 X(1)=B(1)/A(1,
1)
260 GOTO 720
270 Z=1
280 GOTO 720
290 M=N-1
300 FOR I=1 TO M
310 BC=ABS(A(I,I))
320 L=I
330 IJ=I+1
340 FOR J=IJ TO N
350 IF ABS(A(J,I))
<BC THEN 380
360 BC=ABS(A(J,I))
370 L=J
380 NEXT J
390 IF BC<>0 THEN
420
400 Z=1
410 GOTO 720
420 IF L=I THEN 51
0
430 FOR J=I TO N
440 G=A(L,J)
450 A(L,J)=A(I,J)
460 A(I,J)=G
470 NEXT J
480 G=B(L)
490 B(L)=B(I)
500 B(I)=G
510 FOR J=IJ TO N
520 T=A(J,I)/A(I,I
)
530 FOR K=IJ TO N
540 A(J,K)=A(J,K)-
T*A(I,K)
550 NEXT K
560 B(J)=B(J)-T*B(
I)
570 NEXT J
580 NEXT I
590 IF A(N,N)<>0 T
HEN 620
600 Z=1
610 GOTO 720
620 X(N)=B(N)/A(N,
N)
630 I=N-1
640 S=0
650 IJ=I+1
660 FOR J=IJ TO N
670 S=S+A(I,J)*X(J
)
680 NEXT J
690 X(I)=(B(I)-S)/
A(I,I)
700 I=I-1
710 IF I>0 THEN 64
0
720 IF Z<>1 THEN 7
50
730 PRINT #PF,"No
Solution Found":IF
 PF=0 THEN PAUSE
740 GOTO 785
750 PAUSE ALL:FOR
I=1 TO N
760 II$=STR$(I):XX
$="X("&II$&")= "
770 PRINT #PF,XX$;
X(I)
780 NEXT I
785 IF PF=1 THEN P
RINT #1
790 STOP
800 IF PF=1 THEN C
LOSE #1:PF=0
810 INPUT "Use pri
nter? Y/N ";N$
820 IF N$="Y"OR N$
="y"THEN PF=1 ELSE
 890
830 PRINT "Device
Numbers:":PAUSE 2
840 PRINT "For the
 HX-1000 enter 10"
:PAUSE 2
850 PRINT "For the
 PC-324 enter 12":
PAUSE 2
860 INPUT "Enter d
evice number ";D$
870 OPEN #1,D$,OUT
PUT
880 IF D$="10"THEN
 PRINT #1,CHR$(18)
890 PRINT #PF:RETU
RN
999 END
```

Another Simultaneous Equation Solution - (cont)

One of the important features of the Swinnen books is the provision of example problems for each program. The two examples for the simulteneous equations program are:

12a + 22b + 33c = 15	and	b - 2c = -8
23a + 34b + 56c = 18		b + c = 7
a + 2b + 3c = 7		2b - c = 10

The exact solution for the first example is a = -62, b = -46.8 and C = 54.2 . Sample printouts for the first problem with both the PC-324 and the HX-1000 are illustrated at the right. The apparently exact solution from the program is a result of the ten digit output format of the print statement. If PRINT USING with the format ###.########## were used the printout would be:

```
-62.0000000034
-46.8000000027
 54.2000000030
```

where those values are about 500 times more accurate than the values shown in the book for one of the Sharp machines.

```
Simultaneous Equations

A(1,1)=  12
A(1,2)=  22
A(1,3)=  33
B(1)=  15

A(2,1)=  23
A(2,2)=  34
A(2,3)=  56
B(2)=  18

A(3,1)=  1
A(3,2)=  2
A(3,3)=  3
B(3)=  7

X(1)= -62.
X(2)= -46.8
X(3)=  54.2
```

```
Simultaneous Equations

A(1,1)=  12
A(1,2)=  22
A(1,3)=  33
B(1)=  15

A(2,1)=  23
A(2,2)=  34
A(2,3)=  56
B(2)=  18

A(3,1)=  1
A(3,2)=  2
A(3,3)=  3
B(3)=  7

X(1)= -62.
X(2)= -46.8
X(3)=  54.2
```

A more demanding test of a simultaneous equation solver is the 7x7 sub -Hilbert proposed by George Thomson in V8N6P18 together with the test of the solution to the sub-Hilbert proposed by James Walters in V9N2P18. The results for this program were:

Exact	Solution	Walters Test
56	56.00068891	1.0000000000
-1512	-1512.016156	0.9999999970
12600	12600.119	1.0000000020
-46200	-46200.391	0.9999999973
83160	83160.63758	0.9999999980
-72072	-72072.50508	0.9999999973
24024	24024.15504	0.9999999983
Max Error	1.23E-05	3.0E-09
RMS Error	9.07E-06	2.2E-09

The shorter row reduction program from V8N6P20 yields an RMS error for the solution of 2.57E-06 and an RMS error for the Walters test of 1.43E-08. The errors from the program from the Mathematica Library are five times larger for the solution and six times smaller for the Walters test. Either program yields results which compare favorably with results from other programs and machines. The row reduction program presented in V8N6P20 was incorporated into the the least square programs on V12N1P14 and V12N2P21.

Another Simultaneous Equation Solution - (cont)

The second example in the book is a test of the ability of the program to recognize indeterminate sets of equations. The printout at the upper right illustrates the message "No Solution Found" for this problem.

Note that you must enter the zero coefficients in the example. Clearly, the determinant of the matrix of coefficients is zero. What about other cases where the determinant is zero such as

$$\begin{aligned} 12a + 22b + 33c &= 15 \\ 1a + 2b + 3c &= 7 \\ 2a + 4b + 6c &= 14 \end{aligned} \qquad \text{and} \qquad \begin{aligned} 12a + 22b + 33c &= 15 \\ 1a + 2b + 3c &= 7 \\ 2a + 4b + 6c &= 21 \end{aligned}$$

where we recognize that the system at the left has many solutions and the system at the right has no solution. The middle and lower printouts at the left show the results from the program. How do other linear equation solution programs respond to these indeterminate problems?

The Matrices (MAT) program in the TI-74 Mathematics Library gives the message "THE SYSTEM IS SINGULAR" for both problems.

The row reduction program from V8N6P20 yields the message "... Division by Zero" for all three problems where the determinant should be zero.

The Inversion/Linear Systems program in the TI-95 Mathematics Library yields the message "SINGULAR" for the second problem from the book, and the determinant is zero. The determinant for the matrix of coefficients for the two problems above is 1.2e-12, not zero. The problem at the left yields the solution -62, -50.5 and 56.66666667 which is different from that from the program from the book, but is also a solution. The problem at the right yields -132.0833333, -1.75e13 and 1.166667e13 which is not a solution.

The ML-02 program in the TI-59 Master Library yields zero for the determinant for the second problem from the book. The determinant for the matrix of coefficients for the two problems above is -1.2e-12, not zero. The problem at the left yields the solution -62, -60.5 and 63.33333333 which is different from that from the program from the book or from the TI-95 Mathematics Library, but is also a solution. The problem at the right yields -123.75, -1.75e13 and 1.166667e13 which is not a solution.

```
Simultaneous Equations

A(1,1)=  0
A(1,2)=  1
A(1,3)= -2
B(1)= -8

A(2,1)=  0
A(2,2)=  1
A(2,3)=  1
B(2)=  7

A(3,1)=  0
A(3,2)=  2
A(3,3)= -1
B(3)=  10

No Solution Found
```

```
Simultaneous Equations

A(1,1)=  12
A(1,2)=  22
A(1,3)=  33
B(1)=  15

A(2,1)=  1
A(2,2)=  2
A(2,3)=  3
B(2)=  7

A(3,1)=  2
A(3,2)=  4
A(3,3)=  6
B(3)=  14

X(1)= -62
X(2)=  .5
X(3)=  22.66666667
```

```
Simultaneous Equations

A(1,1)=  12
A(1,2)=  22
A(1,3)=  33
B(1)=  15

A(2,1)=  1
A(2,2)=  2
A(2,3)=  3
B(2)=  7

A(3,1)=  2
A(3,2)=  4
A(3,3)=  6
B(3)=  21

X(1)= -82.08333333
X(2)= -7.E+12
X(3)=  4.666667E+12
```

FIVE FUNCTION CURVE FIT - This program was written in response to requests for a TI-74 curve fit program which would be versatile, but easier to use then some of the recently published programs. I decided to model the program after the Forecasting - Auto Curve Choice program in the Real Estate/Investment Solid State module for the TI-59. That program tests the capability of four functions--linear, power, exponential and logarithmic--to fit the input data, selects the function which yields the largest coefficient of determination (r^2), and provides the ability to calculate values for y as a function of x using the selected best fit function. This program provides the same capability, adds the hyperbolic function, selects the best fit based on the largest magnitude of the correlation coefficient (r), and saves the input data pairs so that the residual errors can be examined. That capability is particulary helpful in identifying wild data points. The program also provides for operator intervention to select one of the functions for fitting, a capability which was also available in the Real Estate/Investment module.

The program includes a fairly complete set of prompts. Users are cautioned that the use of zero or negative input values will cause the "Find Best Fit" option to abort. If non-positive input values are required for the fit then individual functions may be used according to the following table:

Linear	No limits on input values.
Exponential	Y input must be positive.
Power	X and Y input must be positive.
Logarithmic	X input must be positive.
Hyperbolic	X input may not be zero.

In some cases the user may find that a rule against zero values can be circumvented by replacing the zero by a very small positive number.

The sample printout at the right was made with the HX-1000. You may recognize the test problems as being from Maurice Swinnen's Statistics Library book. When run without a printer the program pauses to permit the user to view the output.

The listing on page 25 was printed with the PC-324.

In the next issue I will try to present a TI-95 equivalent.

```
Curve Fitter

X =  1
Y =  3.2
X =  2
Y =  7.4
X =  3
Y =  12
X =  4
Y =  16.8
X =  5
Y =  22

Find the Best Fit

Solution for y = a + bx

a    = -1.82
b    =  4.7
r    =  .9992132427
mean =  0
rms  =  .2038181192

Solution for y = ae^bx

a    =  2.48364533
b    =  .4875082173
r    =  .972298671
mean = -.1009109783
rms  =  1.905187914

Solution for y = ax^b

a    =  3.211745293
b    =  1.180427406
r    =  .9999948273
mean = -.0001312811
rms  =  .0427730287

Solution for y = a + bLn(x)

a    =  1.481268800
b    =  11.27808285
r    =  .9837128812
mean =  8.E-12
rms  =  1.776788182

Solution for y = a + b/x

a    =  21.94822175
b    = -20.20880576
r    = -.8848129877
mean = -1.08E-11
rms  =  3.808229048

Best fit is y = ax^b

Residuals for y = ax^b

d1  = -.011745293
d2  =  .0308833082
d3  =  .044188389
d4  = -.0078944769
d5  = -.0280000814

Predict y for y = ax^b

a =  3.211745293
b =  1.180427405

x    =          3
y(x) =  11.9508830
```

```
Curve Fitter

X =  3
Y =  6.5
X =  5
Y =  8.3
X =  7
Y =  9.51
X =  9
Y =  10.4
X =  11
Y =  11.1

Solution for y = a + bLn(x)

a    =  2.081100883
b    =  3.548748323
r    =  .9999910059
mean =  1.38E-11
rms  =  .0085558251

Residuals for y = a + bLn(x)

d1  =  .0023386457
d2  = -.008439279
d3  =  .0071783081
d4  =  .0058293531
d5  = -.0050808994

Predict y for y = a + bLn(x)

a =  2.081100883
b =  3.548748323

x    =          9
y(x) =  10.38417885
```

Five Function Curve Fit - (cont)

Program Listing:

```
10 A$="Curve Fitter":PRI
NT A$:PAUSE 1
11 B$(1)="Linear      "
:C$(1)="y = a + bx"
12 B$(2)="Exponential  "
:C$(2)="y = ae^bx"
13 B$(3)="Power       "
:C$(3)="y = ax^b"
14 B$(4)="Logarithmic  "
:C$(4)="y = a + bLn(x)"
15 B$(5)="Hyperbolic   "
:C$(5)="y = a + b/x"
16 B$(6)="Find the Best
Fit"
20 DIM X(50),Y(50)
25 INPUT "Use printer? Y
/N ";Z$
30 IF Z$="Y"OR Z$="y"THE
N PN=1 ELSE 70
35 PRINT "Device Numbers
:":PAUSE 1
40 PRINT "For the HX-100
0 enter 10":PAUSE 1
45 PRINT "For the PC-324
 enter 12":PAUSE 1
50 INPUT "Enter device n
umber ";D$
55 OPEN #1,D$,OUTPUT
60 IF D$="10"THEN PRINT
#1,CHR$(18)
65 PRINT #1:PRINT #1,A$
70 PRINT #PN
100 PRINT "End Input by
Entering E":PAUSE 1
105 PRINT "Zero or negat
ive inputs will":PAUSE 1
110 PRINT "prevent some
solution options":PAUSE
1
115 N=1
120 X$="X = "
125 Y$="Y = "
130 INPUT X$;XX$:IF XX$=
"E"OR XX$="e"THEN 170
135 INPUT Y$;YY$:IF YY$=
"E"OR YY$="e"THEN 170
140 X(N)=VAL(XX$)
145 Y(N)=VAL(YY$)
150 IF PN=0 THEN 165
155 PRINT #PN,X$;X(N)
160 PRINT #PN,Y$;Y(N)
165 N=N+1:GOTO 130
170 N=N-1
175 PRINT "The Regressio
n Options are:":PAUSE 1
180 FOR I=1 TO 6
185 PRINT STR$(I)&" - "&
B$(I)&C$(I):PAUSE 2
190 NEXT I
200 INPUT "Which Option?
 ";P
205 IF P=6 THEN PRINT #P
N:PRINT #PN,B$(6):P=1:Q=
6
215 S1=0:S2=0:S3=0:S4=0:
S5=0
220 FOR I=1 TO N
225 U=X(I):V=Y(I)
230 IF P=2 OR P=3 THEN V
=LN(V)
235 IF P=3 OR P=4 THEN U
=LN(U)
240 IF P=5 THEN U=1/U
245 S1=S1+U:S2=S2+U*U
250 S3=S3+V:S4=S4+V*V
255 S5=S5+U*V:NEXT I
260 DET=N*S2-S1*S1
265 IF DET=0 THEN PRINT
"determinant = 0":PAUSE
270 A(P)=(S3*S2-S5*S1)/D
ET
275 B(P)=(N*S5-S1*S3)/DE
T
280 IF P=2 OR P=3 THEN A
(P)=EXP(A(P))
290 R(P)=(S5-S1*S3/N)/SQ
R((S2-S1*S1/N)*(S4-S3*S3
/N))
295 Z=1:GOSUB 800
300 PRINT #PN:PRINT #PN,
"Solution for "&C$(P):IF
 PN=0 THEN PAUSE
305 PRINT #PN:PAUSE ALL
310 PRINT #PN,"a    = ";
A(P)
315 PRINT #PN,"b    = ";
B(P)
320 PRINT #PN,"r    = ";
R(P)
325 PRINT #PN,"mean = ";
M(P)
330 PRINT #PN,"rms  = ";
RMS(P)
335 PAUSE 0:PRINT
340 IF Q=6 AND P<5 THEN
P=P+1:GOTO 215
345 IF Q<>6 THEN 400
350 RM=ABS(R(1)):RI=1
355 FOR I=2 TO 5
360 IF ABS(R(I))>RM THEN
 RM=ABS(R(I)):RI=I
365 NEXT I
370 PRINT #PN:PAUSE ALL
375 P=RI
380 PRINT #PN,"Best fit
is "&C$(P)
385 PAUSE 0
400 Y$="Residuals for "&
C$(P)
405 INPUT Y$&" ? ";Z$
410 IF Z$="N"OR Z$="n"TH
EN 500
415 PRINT #PN:PRINT #PN,
Y$:PRINT #PN
420 Z=2
425 GOSUB 800
500 Y$="Predict y for "&
C$(P)
505 INPUT Y$&" ? ";Z$
510 IF Z$="N"OR Z$="n"TH
EN 600
515 PRINT #PN:PRINT #PN,
Y$:PRINT #PN
520 PRINT #PN,"a = ";A(P
)
525 PRINT #PN,"b = ";B(P
)
530 PRINT #PN
535 X$="x    = ":Y$="y(x
) = "
540 INPUT X$;U
545 PAUSE ALL:PRINT #PN,
X$,U
550 V=0:Z=3:GOSUB 815
555 PRINT #PN,Y$;-D
560 PAUSE 0:PRINT #PN
565 INPUT "Try another x
 (Y/N) ? ";Z$
570 IF Z$="Y"OR Z$="y"TH
EN 540
600 INPUT "Try another o
ption (Y/N) ? ";Z$
610 IF Z$="Y"OR Z$="y"TH
EN Q=0:GOTO 175
799 STOP
800 S1=0:S2=0
805 FOR I=1 TO N
810 U=X(I):V=Y(I)
815 ON P GOTO 820,825,83
0,835,840
820 D=V-A(1)-B(1)*U:GOTO
 845
825 D=V-A(2)*EXP(B(2)*U)
:GOTO 845
830 D=V-A(3)*U^B(3):GOTO
 845
835 D=V-A(4)-B(4)*LN(U):
GOTO 845
840 D=V-A(5)-B(5)/U
845 ON Z GOTO 870,850,89
0
850 D$="d"&STR$(I)
855 IF I<10 THEN D$=D$&"
 "
860 D$=D$&" = "
865 PRINT #PN,D$;D:IF PN
=0 THEN PAUSE
870 S1=S1+D:S2=S2+D*D
875 NEXT I
880 M(P)=S1/N
885 RMS(P)=SQR(S2/N)
890 RETURN
999 END
```

MANY DIGIT SQUARE ROOT IN BASIC - Larry Leeds

Larry writes Basic programs on the Radio Shack Model 100 which is a fourteen digit, base 10 machine. Conversion for use on the CC-40 and TI-74, which are 13-14 digit, base 100 machines, is typically straightforward. But this program for calculation of multiprecision square roots was different. The program listing is at the right. Larry's instructions were "Move the decimal point two places at a time until the remaining integer is one or two digits. If one digit, partition with seven digits in the first group, with the remaining groups six digits each. If two digits, partition with six digits in each group. Add trailing zeroes to the last group if needed." In response to the program prompts you should provide twice as many digits for N as for the square root of N.

Line 70 is provided to make it possible to generate square roots of integers without entering all of the trailing zeroes. For $\sqrt{3}$ simply remove the REM indication. For square roots of other integers make a suitable replacement for the 3.E+06 in line 70.

Page 21 includes a short addition to the program which makes it possible to run a test problem which yields an answer of 20 blocks of all nines. With steps 800-870 in place simply enter RUN 800 in the display, press ENTER, and wait about minute for the solution to appear in the display.

The first problems that I tried with this program were square roots of single digit integers. The square roots of 2, 6, and 8 were normal, but the square roots of 3, 5, and 7 yielded negative values in the third block of the output. Tests of the square roots of 2, 3, 5, 6, 7, and 8 for 100 blocks (600 digits) did not show negative values in the output at any other position. The program was modified to work with five digit blocks to see if the problem would be cleared. (V12N2P12 had reported that another program had problems when converted from the Model 100 to the TI-74, and the problems were cleared by changing from six digit blocks to five digit blocks.) The solutions for $\sqrt{3}$ and $\sqrt{7}$ were correct, but the solution for $\sqrt{5}$ continued to have a negative value for the third block. The printouts for the first six blocks of $\sqrt{3}$ and $\sqrt{5}$ are shown below for both the six digit and five digit block programs.

1732050	2236067	173205	223606
807569	977500	08075	79775
-122707	-210304	68877	0-211
527446	409173	29352	69640
341505	668731	74463	91736
872366	276235	41505	68731

The same problem did not appear with the Model 100 implementation. Comparative testing isolated the source of the TI-74 problem to the first statement in line 150. Testing with the statement D=INT(C*2)+1 changed to D=INT(C*2) showed that the negative output problem had been eliminated for the square roots of 3, 5, and 7. See the printouts for $\sqrt{3}$ and $\sqrt{5}$ on the next page.

```
10 DIM N(200)
20 Z=1.E+06:J=1
30 PRINT "*** SQR(N) ***
":PAUSE 2
35 INPUT "Use Printer Y/
N ? ":Z$
40 IF Z$="N"OR Z$="n"THE
N 50
45 PN=1:OPEN #1,"12",OUT
PUT
50 INPUT "Number of grou
ps in Sqr(N) ? ":W
60 INPUT "Number of grou
ps in N ? ":X
70 REM N(1)=3.E+06:GOTO
115
80 FOR I=1 TO X
90 N$="N("&STR$(I)&") =
"
100 INPUT N$:N(I):NEXT I
115 PRINT "wait"
120 A=N(J)*Z^3+N(J+1)*Z^
2
130 B=SQR(A)/Z
140 C=B-INT(B)
150 D=INT(C*Z)+1:B=B-C
160 N(J+1)=N(J)*Z+N(J+1)
-B^2
170 B=B*2:N(J-1)=B
180 N(J)=D:F=W-1:G=W+1
200 H=G-F:L=H:J=H
210 N(J+1)=N(J)*Z+N(J+1)
:N(J)=0:J=J+1
220 J=J+H-1:K=H-1:A=K:Q=
J:C=Z
250 M=N(J)+C-Z-N(K)*D
260 C=M/Z+Z
270 P=C-INT(C)
280 C=C-P:N(J)=P*Z
290 K=K-1:J=J-1:H=H-1
300 IF H<>0 THEN 250
310 IF C=Z THEN 400
320 M=(C-Z)*Z:J=J+1:N(J)
=N(J)+M
330 IF M>=Z THEN 400
340 C=0:D=D-1:H=L:K=A:J=
Q:N(J)=N(J)+D
350 FOR Y=H TO 1 STEP -1
360 C=(N(K)+N(J)+C)/Z
370 M=C-INT(C)
380 N(J)=M*Z:C=C-M:K=K-1
:J=J-1:NEXT Y
390 J=J+1:N(J)=N(J)+C*Z:
N(A)=D
400 H=L-1:K=A:N(K)=N(K)+
D
410 FOR Y=H TO 1 STEP -1
420 IF N(K)<Z THEN 460
430 C=N(K)/Z:M=C-INT(C)
440 N(K)=M*Z:C=C-M:K=K-1
450 N(K)=N(K)+C:NEXT Y
460 F=F-1:IF F<>0 THEN 6
00
470 H=L:K=0:C=0
490 N(K)=(N(K)+C)/2
500 M=N(K)-INT(N(K))
510 N(K)=N(K)-M:C=M*2*Z
520 K=K+1:H=H-1:IF H<>0
THEN 490
530 PAUSE ALL:FOR I=0 TO
 W-1
540 L=6-LEN(STR$(N(I))):
IF L<0 THEN L=0
550 N$=RPT$("0",L)&STR$(
N(I))
560 PRINT #PN,N$:NEXT I:
END
600 K=A+2
610 M=((N(K)*Z+N(K+1))*Z
+N(K+2))*Z+N(K+3)
615 K=K+3
620 S=((N(0)*Z+N(1))*Z+N
(2))
630 T=M/S:D=INT(T):A=A+1
:N(A)=D
640 GOTO 200
```

Many Digit Square Root in Basic - (cont)

1732050
807568
877293
527446
341505
872366

2236067
977499
789696
409173
668731
276235

```
800 W=21:X=40:J=1:Z=:.E+
06
810 FOR K=1 TO 19
820 N(K)=999999:NEXT K
830 N(20)=999998
840 FOR K=21 TO 39
850 N(K)=0:NEXT K
860 N(40)=1
870           GOTO 115
```

No other problems have been identified to date with this change in place. Furthermore, no problem has been identified when the same change is made to the Model 100 program. Even so, we were uneasy.

Myer Boland ran the program on his TI-99/4 which has the same 13/14 digit, base 100 mechanization as the TI-74 and CC-40. He found the same negative value problem in the third block of the output. His solution was to change the first part of the user instructions to "Move the decimal point two places at a time until the remaining integer is one or two digits. If one digit, partition with five digits in the first group, with the remaining groups six digits each. ... ". Thus, to solve for the square root of three, the first input block would be 30000. No problems have been identified with this method on either the TI-99/4, the TI-74 or the Model 100. The printouts for the square roots of 2, 3, 5, 6, 7 and 8 are:

141421	173205	223606	244948	264575	282842
356237	080756	797749	974278	131106	712474
309504	887729	978969	317809	459059	619009
880168	352744	640917	819728	050161	760337
872420	634150	366873	407470	575363	744841
969807	587236	127623	589139	926042	939615

In a sense the negative output is correct. Consider the square root of five solution where the second and third lines are 977500 and -210304. If a one is borrowed from the second line it becomes 977499. If the magnitude of the third line is subtracted from the borrowed 1000000 the third line becomes 789696. The modified lines are correct. This suggests another solution for the negative output problem: simply scan the solution before displaying or printing it. If any group is negative perform the borrowing process outlined above.

Although the negative output has only appeared in the third group of any solution that has been tried to date some experimentation will show that negative values have appeared in other groups while the program is running. This ability to reach "correct" solutions even if some of the elements become negative suggests that the algorithm is particularly forgiving. Can one of our members explain?

BOOK REVIEW - TI-74 BASICALC Technical Data Manual. This 30 page manual was announced on page 8 of the latest issue of Programmable Calculator News. It is available from TI for ten dollars. Call 1-800-TI-CARES for the latest information on ordering.

The manual includes descriptive text, block diagrams, schematics, memory maps, and interface details. A must document for those who want to delve deeply into the workings of the TI-74.

MULTIPLE LINEAR REGRESSION WITH TWO VARIABLES - P. Hanson. I needed a linear regression with two variables for an application where I work. I modified a Model 100 program for regression with user defined functions similar to that on V12N1P14. I needed a sample problem to check the capability of the modified program and remembered that there was an equivalent program and a sample problem in the Statistics Library book reviewed on page 4 of this issue. When I entered the values from the table on page 50 of the book I got answers that were different from those listed on page 51 of the book. I knew that the different numerical precision of the Model 100 and the Sharp machines and the different methods of solution would lead to somewhat different results, but the actual difference seemed too large. I then converted the program from the book for the Model 100, entered the sample problem from page 50 of the book, and got an answer which agreed with the result with my modified version of the user defined function program to seven decimal places. After a considerable amount of rechecking of programs and data entries I found that there was a typographic error in the table in the book. The first y value in the table was listed as -9.9 when it should have been -9.99.

One thing that this exercise did prove is how versatile the program from V12N1P14 is. The only changes required to accept two independent variables W and X, and one dependent variable Y are:

1. Add W(50) in line 100.

2. Add the following lines:

```
143  A$="W"&STR$(I)&" = ":INPUT A$;W(I)

146  IF PN<>0 THEN PRINT #PN,A$,W(I)
```

3. For a linear regression with two variables enter the following user defined functions at the end of the program:

```
810 F(1)=1
820 F(2)=W(L)
830 F(3)=X(L)
840 RETURN
```

The printout from a modified TI-74 program for the sample problem from the Statistics Library book appears at the right. Some other solutions for the same sample problem are:

Sharp from the book, pages 50-51	53.55751069	-12.08011976	0.1317365269
Model 100, book method	53.5575106929	-12.080119760479	0.13173652694611
Model 100, converted from V12N1P14	53.53750998009	-12.080119522791	0.13173647935745

```
W1 =          1
X1 =      -9.99
Y1 =      40.16

W2 =          2
X2 =      -4.98
Y2 =      28.74

W3 =          3
X3 =          0
Y3 =      17.32

W4 =          4
X4 =       5.01
Y4 =        5.9

W5 =          5
X5 =       9.98
Y5 =      -5.53

W6 =          6
X6 =         15
Y6 =     -16.95

W7 =          7
X7 =      19.98
Y7 =     -26.37

A1 =  53.55750988
A2 = -12.08011949
A3 =  .1317364727

d1 = -.0013430286
d2 = -.0012232672
d3 =  .0028485882
d4 =  .0029683496
d5 = -.0016424302
d6 = -.0028400336
d7 =  .0012318219

Mean =  4.285714E-13

S.E. =  .0028507199
```

On the next page we will see a more dramatic illustration of the differences between solutions on different machines.

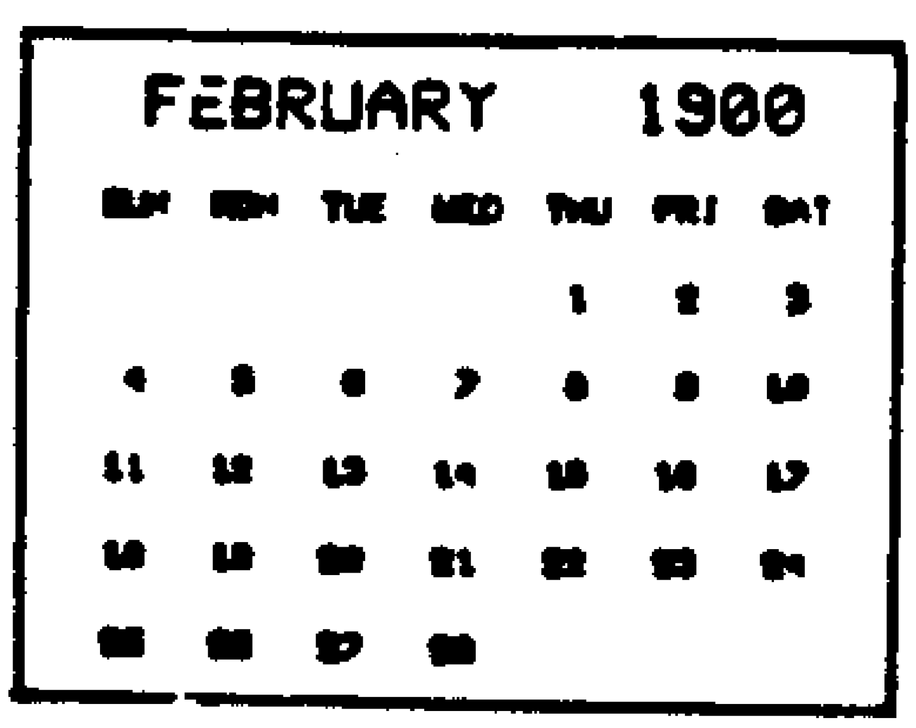
CALENDAR PROGRAM FOR THE TI-74 - P. Hanson.

Long time members know that I am fascinated by calendar printing programs, and so will not be surprised that I have written one for the TI-74. In this case the program also serves to illustrate the differences in the printing capability of the TI-74/PC-324 and the CC-40/HX-1000.

A calendar program for the CC-40/HX-1000 appeared in V9N5P8. Allowing two characters for each date and one space between dates yields 20 characters as the minimum number needed per line. That was the format used on the TI-59/PC-100 programs where the PC-100 provided exactly 20 characters per line. The HX-1000 provided two print modes, either 18 characters per line or 36 characters per line. One solution would have been to use the 18 character per line mode and the European format as in Dave Leising's calendar program for the TI-66/PC-200. I chose another alternative: the program in V9N5P8 used the 18 character mode for the month and year, and switched to the 36 character mode for the days of the week and the dates. Three-letter abbreviations were used for the days of the week. The actual program was a minimum change version of an earlier program I had written to display a calendar on the screen of my Model 100. A sample printout from the CC-40/HX-1000 program appears below. The use of two character sizes provides an attractive format. Printout of a single month required about 21 seconds, agonizingly slow by TI-59 standards where Patrick Acosta's program in V9N2P7 prints out a a full year in only 83 seconds when running in fast mode.

The program at the right is a modification of the CC-40 program for use with the TI-74/PC-324. The necessary modification accommodates the 24 character print width of the PC-324 by returning to a format similar to that used with the TI-59/PC-100, that is each column separated from the next by only one space, and two-letter abbreviations for the days of the week. I also cleaned up the program by removing the subscript notation for the individual lines of the calendar. I can't recall why I used subscripts in the first place. A representative printout appears below. A month can be printed in nine seconds, about the same speed as the TI-59/PC-100 in fast mode, and much faster than the CC-40/HX-1000.

CC-40/HX-1000



TI-74/PC-324

```
FEBRUARY          2000
Su Mo Tu We Th Fr Sa
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29
```

```
100 DIM Q(12)
105 DATA 31,28,31,30,31,
30,31,31,30,31,30,31
110 FOR I=1 TO 12:READ Q
(I):NEXT I
115 DATA "JANUARY  ","FE
BRUARY ","MARCH    ","AP
RIL    ","MAY      ","JU
NE     "
120 DATA "JULY     ","AU
GUST   ","SEPTEMBER","OC
TOBER  ","NOVEMBER ","DE
CEMBER "
130 INPUT "Enter Month (
1-12): ";M
135 IF M<1 OR M>12 THEN
130
140 INPUT "Enter Year (>
1582): ";R
145 IF R<1583 THEN 140
150 IF R-4*INT(R/4)=0 TH
EN Q(2)=29
155 IF R-100*INT(R/100)=
0 THEN Q(2)=28
160 IF R-400*INT(R/400)=
0 THEN Q(2)=29
165 R1=R-1:R2=R+INT(R1/4
)-INT(R1/100)+INT(R1/400
)
170 FOR I=0 TO M-1:R2=R2
+Q(I):NEXT I
175 D1=R2-7*INT(R2/7)
180 RESTORE 115:FOR I=1
TO M:READ M$:NEXT I
185 OPEN #1,"12",OUTPUT
190 PRINT #1,TAB(3);M$;"
     ";R
210 PRINT #1," Su Mo Tu
 We Th Fr Sa"
225 C$=" "&RPT$("   ",D1
)
230 FOR I=1 TO 7-D1:C$=C
$&"  "&STR$(I):NEXT I
235 PRINT #1,C$
245 I=8-D1
250 C$=" "
255 FOR L=1 TO 7
260 IF I>9 THEN B$=" " E
LSE B$="  "
265 C$=C$&B$&STR$(I)
270 I=I+1:IF I>Q(M)THEN
PRINT #1,C$:GOTO 300
275 NEXT L
280 PRINT #1,C$
285 GOTO 250
300 PRINT #1
305 CLOSE #1:GOTO 130
```