Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

Materials & Molecular
Research Division

HP-41C CALCULATOR PROGRAMS FOR FITTING OF DATA BY AN ANALYTICAL FUNCTION

Leo Brewer

December 1982

Prepared for the U.S. Department of Energy under Contract DE-AC03-76SF00098

LBL-15346

LBL--15346

DE83 006441

HP-41C CALCULATOR PROGRAMS FOR FITTING OF DATA BY AN ANALYTICAL FUNCTION

by

Leo Brewer

Materials and Molecular Research Division, Lawrence Berkeley Laboratory
and Department of Chemistry
University of California, Berkeley, California 94720

This work was supported by the Division of Materials Sciences, Office of Basic Energy Sciences, U.S. Department of Energy, Under Contract No. DE-AC03-76SF00098

This manuscript was printed from originals provided by the author.

Introduction

The general availability of programmable calculators and computers has signaled a shift from the tabular presentation of thermodynamic data to presentation in the form of analytical equations and the replacement of graphical methods of treating data by analytical methods. HP-65 and HP-67 calculator programs for a variety of thermodynamic calculations have been presented in two earlier reports.[1,2] The present report lists programs for the HP-41c calculator which are of particular use in representing thermodynamic data in analytical form.

The first section will deal with analytical equations for interpolation purposes. The equations are fit to two, three, or four tabulated points. In particular, the values of $-(G^o-H^o_{Std})/RT$ for the reactants and products of a reaction are combined to yield an equation for $-\Delta(G^o-H^o_{Std})/RT$ which can then be used to obtain values of the equilibrium constant at desired temperature by the relation

$$\ln K = -\Delta G^o/RT = -\Delta(G^o-H^o_{Std})/RT - \Delta H^o_{Std}/RT$$

This procedure is much simpler than the use of ΔS^o_T and ΔH^o_T to obtain ΔG^o_T as the contribution of ΔC^o_P causes more rapid changes of ΔS^o and ΔH^o with temperature than of $-\Delta(G^o-H^o_{298})/RT$, for which the ΔC_P contributions largely cancel.

The second section deals with fitting $-(G^o-H^o_{Std})/RT$ values or other quantities tabulated for a large number of evenly spaced temperatures. A least-square fit is provided using Chebyshev orthogonal polynomials.

The third section deals with least-square fitting of sets of x,y values to an equation for f(y) given as either $a + bf(x)$, $af_1(x) + bf_2(x)$, $a + bf_1(x) + cf_2(x)$, or $af_1(x) + bf_2(x) + cf_3(x)$. Some examples are given of the types of functions needed for typical thermodynamic calculations.

The initial sections give the directions for using the calculator programs and the steps of the program are listed along with use of the storage registers. In an appendix, a more detailed discussion is given in regard to the reasons for the procedures used and operation of the program. Possible modifications of the programs for special purposes are also listed. This report is subdivided as follows:

page

Program Steps

CHAPTER I

Interpolation Fit to $y = \Sigma a_n x^n$

Program INTERP will fit a pair of x,y values to a linear equation, a set of three x,y values to a quadratic equation, or four x,y pairs with the x values at evenly spaced intervals of magnitude I to a cubic polynomial. In addition, the program is designed to accept values of $-(G^o-H^o_{Std})/RT$ of $-(G^o-H^o_{Std})/T$ for each of the reactants and products of a chemical reaction at two, three, or four temperatures and fit the resulting $-\Delta(G^o-H_{Std})/RT$ values to an interpolation equation which can be combined with $\Delta H^o_{Std}/R$ for the reaction to calculate lnK or K, the equilibrium constant of the reaction, at desired temperatures in the interpolation range.

Directions:

(1) Insert tape INTERP or XEQ INTERP if program already inserted but calculator is positioned on another program.

	2 Pt. Fit		Display
(2a)	$y_1 \uparrow y_2$	XEQ2	y_1-y_2
(3a)	$x_1 \uparrow x_2$	R/S	a_0
(4a)		SST	a_1
(5)	x	User E	$\hat{y}$
	3 Pt. Fit		
(2b)	$y_1 \uparrow y_2 \uparrow y_3$	XEQ 3	y_1-y_2
(3b)	$x_1 \uparrow x_2 \uparrow x_3$	R/S	a_0
(4b)		SST SST	a_1, a_2
(5)	x	User E	$\hat{y}$
	4 Pt. Fit		
(2c)	$y_1 \uparrow y_2 \uparrow y_3 \uparrow y_4$	XEQ4	$I^3 a_3$
(3c)	$I \uparrow x_1$	R/S	a_0

4 Pt. Fit (cont.)		Display
(4c)	SST SST SST	a_1, a_2, a_3
(5) x	User E	$\hat{y}$

For values of x spaced at a regular interval I, R/S will give y(x+I) after calculation of y(x) if I is in register 11.

For the reaction aA + bB = mM + nN, the values of $g = (-G^o - H^o_{Std})/RT$ (a positive number) for the reactants and products are keyed in as follows:

		Display
(a) n ↑ m ↑ -b ↑ -a The sign is positive for products and negative for reactants. For total of three reactants and products, n=0. 0 ↑ 0 ↑ m ↑ -a if only two.	XEQ 1	n
(b) g_N ↑ g_M ↑ g_B ↑ g_A Repeat (b) for each temperature.	A	ng_N
(c)	XEQ 10	$-\Delta(G^o - H^o_{Std})/RT_4$

(d_4) For 4 pt. fit, XEQ 4 and continue with 3c and 4c.

(d_3) For 3 pt. fit, SST XEQ 3 and continue with 3b and 4b.

(d_2) For 2 pt. fit, SST SST XEQ 2 and continue with 3a and 4a.

(6) XEQ 6 to divide by R if $-(G^o - H^o_{Std})/T$ values were inserted in steps (b) and R is stored in reg. 4.

(7) To obtain $\Delta H^o_{Std}/R$ of reaction, key 4.1 STO6 and then enter $\Delta H^o_{298}/R$ for each product and reactant as in (b).

(8a) To calculate $\Delta H^o_{298}/R$ from a set of ln K_T values, XEQ 7 is followed by

T ↑ ln K_T Step (8a) is repeated for all T.	User H	$\Delta H^o_{298}/R$
(8b)	R/S SST	av $\Delta H^o_{298}/R$, Std. Dev.
(9) T	XEQ 5, SST	ln K, K

Note 1: Steps (b) and (c) will accomodate a total of four reactants and products without any modification. Additional reactants and products can be accomodated by c ↑ g_C X ST+IND6 following step (b) and similarly for ΔH after step (7).

Note 2: The program can be used for $-(G^o - H^o_{Std})/T$ and ΔH^o_{Std} as well as for the dimensionless quantities used to illustrate the displays, but step 9 will display R lnK instead of lnK and it must be divided by R before obtaining

K. R in appropriate units can be stored in register 4 for use in step (6) to convert the equation for $-\Delta(G^{o}-H^{o}_{Std})/T$ to the dimensionless $-\Delta(G-H^{o}_{Std})/RT$ form; so it is unnecessary to divide by R each time step 9 is carried out. Of course, the appropriate ΔH^{o}_{Std} or $\Delta H^{o}_{Std}/R$ must be used.

Note 3: The values of $-\Delta(G^{o}-H^{o}_{Std})/RT$ obtained at each temperature are stored starting with R18. Thus a set of values at three or more temperatures is available for repeat fits using only a portion of the values.

```
01♦LBL "INTERP"
02♦LBL 10
RCL 18  RCL 19  RCL 20
RCL 21  RTN  RDN  RDN

10♦LBL 02
STO 12  -  0  STO 02
STO 03  RDN  RTN  STO 11
-  /  STO 01  RCL 11  *
CHS  RCL 12  +  STO 00
RTN  RCL 01

30♦LBL 03
X<>Y  -  STO 13  RDN
LASTX  STO 12  -  0
STO 02  STO 03  RDN  RTN
STO 15  RDN  STO 11  RDN
STO 14  RCL 11  -  /
RCL 11  RCL 15  +
STO 17  *  RCL 14
RCL 11  +  RCL 13  *
RCL 15  RCL 11  -
STO 16  /  -  RCL 15
RCL 14  -  /  STO 01
RCL 13  RCL 16  /  -
CHS  RCL 17  /  STO 02
RCL 11  XEQ E  CHS
RCL 12  +  STO 00  RTN
RCL 01  RCL 02

89♦LBL 04
R↑  STO 12  -  STO 15
RDN  LASTX  -  STO 14
RDN  LASTX  -  STO 13
RCL 15  3  /  +
RCL 14  -  2  /  RTN
STO 16  RDN  STO 11  3
Y↑X  /  STO 03  RCL 16
RCL 11  +  *  3  *  CHS
RCL 14  RCL 13  2  *  -
2  /  RCL 11  X↑2  /  +
STO 02  RCL 13  RCL 11
/  RCL 11  RCL 16  2  *
+  RCL 02  *  -  RCL 11
RCL 16  +  RCL 16  *  3
*  RCL 11  X↑2  +
RCL 03  *  -  STO 01  0
STO 00  RCL 16  XEQ E
CHS  RCL 12  +  STO 00
RTN  RCL 01  RCL 02
RCL 03

174♦LBL 05
XEQ E  RCL 05  R↑  /  -
RTN  E↑X

182♦LBL E
ENTER↑  ENTER↑  ENTER↑
RCL 03  *  RCL 02  +  *
RCL 01  +  *  RCL 00  +
RTN  RDN  RCL 11  +
GTO E

201♦LBL 01
STO 07  RDN  STO 08  RDN
STO 09  RDN  STO 10
17.1  STO 06  RDN  RTN

213♦LBL A
ISG 06  STO IND 06  CLX
RCL 07  ST* IND 06  RDN
RCL 08  *  ST+ IND 06
RDN  RCL 09  *
ST+ IND 06  RDN  RCL 10
*  ST+ IND 06  RTN

232♦LBL 07
ΣREG 12  CLΣ  RTN

236♦LBL H
STO 14  RDN  XEQ E
RCL 14  -  R↑  *  Σ+
LASTX  RTN  MEAN  STO 05
RTN  SDEV

251♦LBL 06
RCL 04  ST/ 00  ST/ 01
ST/ 02  ST/ 03  .END.
```

257 steps SIZE 022

309 bytes

<u>Test:</u>

(2b) 1.978 ↑ 2.536 ↑ 3.25 XEQ 3 -0.558;

(3b) 0.3 ↑ 0.4 ↑ 0.5 R/S 1.240;

(4b) SST 0.120 SST 7.800; (5) 0.4 E 2.536;

(2c) 1.552 ↑ 1.978 ↑ 2.536 ↑ 3.25 XEQ 4 0.004;

(3c) 0.1 ↑ 0.2 R/S 1.000

(4c) SST 2.000 SST 3.000 SST 4.000; (5) 0.4 E 2.536

$C(gr) + 2Cl_2(g) = CCl_4(s)$, (a) 0 ↑ 1 ↑ -2 ↑ -1 XEQ1, 0

500	(b_1)	68.1 ↑ 49.85 ↑ 1.16	A 0
1000	(b_2)	81.31 ↑ 55.43 ↑ 2.78	A
1500	(b_3)	90.01 ↑ 58.85 ↑ 4.19	A
2000 K	(b_4)	96.53 ↑ 61.34 ↑ 5.38	A

(c) XEQ 10, -31.53; (d_4) XEQ 4, -0.02; (3c) 500 ↑ 500 R/S - 33.05;
(4c) SST 3.60×10^{-4} SST 5.20×10^{-7} SST 1.60×10^{-10}; (5) EEX 3 E -32.33;
1.98719 STO 4, $\Delta H^\circ_o = -25 \times 10^3$ RCL 4 /=-12581 STO 5; XEQ 6, 1.987

	lnK		K
(9) 500 XEQ 5	8.676	SST	5858
750 XEQ 5	0.392	"	1.479
EEX3 XEQ 5	-3.689	"	2.50×10^{-2}
1500 XEQ 5	-7.656	"	4.73×10^{-4}
2EEX3 XEQ 5	-9.576	"	6.94×10^{-5}

XEQ 10, -31.53; SST -31.88; (2b) XEQ 3, -0.43;
(3b) 500 ↑ EEX 3 ↑ 1500 R/S -33.17; (4b) SST 8.0×10^{-4} SST 4.0×10^{-8},
XEQ 6, 1.987

(9) 500 XEQ 5 8.676, SST 5858
750 XEQ 5 0.395, SST 1.485
1500 XEQ 5 -7.656, SST 4.73×10^{-4}

XEQ 10 -31.53; SST -31.88; SST -32.33; (2a) XEQ 2, -0.43;
(3a) 500 ↑ EEX 3 R/S -33.19; (4a) SST 8.6×10^{-4} (5) EEX 3 E -32.33;
XEQ 6, 1.987

(9) 750 XEQ 5 0.397, SST 1.487
1500 XEQ 5 -7.666, SST 4.69×10^{-4}

R	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	a_0	a_1	a_2	a_3	R	$\frac{\Delta H_{Std}}{R}$	Index	-a	-b	m	n	x_2	y_2					
												x_2	y_2	y_3-y_2	x_1	x_3	(x_3-x_2)	(x_2+x_3)
												I	y_1	y_2-y_1	(y_3-y_1)	(y_4-y_1)	x_1	

$$\overbrace{-\Delta(G^{\circ}-H^{\circ}_{Std})/RT \text{ for 2 to 4 temp.}}^{18 \qquad 19 \qquad 20 \qquad 21}$$

R	12	13	14	17
For steps 8a and 8b	$\sum\frac{\Delta H^{\circ}_{298}}{R}$	$\sum\left(\frac{H^{\circ}_{298}}{R}\right)^2$	lnK	n

The minimum SIZE is 022. If data for more than four temperatures are used in steps (b) and (c), the values of $-\Delta(G^{\circ}-H^{\circ}_{298})/RT$ will be stored in R22 and beyond if a larger SIZE is specified.

CHAPTER II

Data Fitting Using the Chebyshev Polynomials

The Chebyshev (Tschebycheff) polynomials, $T_n(x) = \cos(n\cos^{-1}x)$, are orthogonal over the continuous interval $0 \leq x \leq 1$ and they have been shown to be the most economical polynomial for expressing f(x) as a polynomial series with the minimum number of terms for a given accuracy.[3,4] The Chebyshev polynomial can be modified to $C_n(\bar{x})$ which is orthogonal for discrete integer values of the variable, $\bar{x}$, from 0 to N as discussed in references 1-4. If x_i is the initial value of x and I is the regular interval between x values, the data points are assigned integral $\bar{x}$ values from 0 to N where $\bar{x} = (x-x_i)/I$ and the data are fit to a polynomial of the type

$$f(\bar{x}) = c_0C_0(\bar{x}) + c_1C_1(\bar{x}) + c_2C_2(\bar{x}) + c_4C_4(\bar{x}).$$

Because of the orthogonality of $C_n(\bar{x})$, the matrix calculations for the least-square fit of the data are simple and the C_n constants do not depend upon whether the quartic term is included or not. Also, the symmetry of the function reduces the calculations by half. An additional advantage is the more symmetrical weighting over a wide range of data points than for many other fitting procedures. After fitting of the equation, the value of the quartic term for $\bar{x} = 0$ or N is displayed and a decision is made whether the quartic term is large enough to retain. Then the equation is transformed to a power series of third or fourth order:

$$f(\bar{x}) = \alpha_0 + \alpha_1\bar{x} + \alpha_2\bar{x}^2 + \alpha_3\bar{x}^3 \ (+ \alpha_4\bar{x}^4)$$

and

$$f(x) = a_0 + a_1x + a_2x^2 + a_3x^3 \ (+ a_4x^4).$$

The instructions for use of the program are given followed by a listing of the program steps. For a more detailed record of the various operations of the program and the reasons for the procedures used, an Appendix is provided which gives equations used, the general flow chart and discusses the indices and subroutines used.

Directions for Fitting N+1 Data Points at Even I Intervals of x

(1) If programs are not already stored in calculator, insert Prgm. CH card (2 sides) and Prgm. CB cards (7 sides).

(2a) If the $C_n(\bar{x})$ values for N+1 data points have not already been calculated and stored, key

N+1 XEQ 'CH' → SIZE? = > 3N+20(N odd) or 3N+22(N even)

F1 will be set if N is even. If number of storage registers is not adequate, key

XEQ 'SIZE' mno where mno is three digit number corresponding to the number of registers needed.

(2b) R/S → Index = 21 + q/1000 for C_n values in R21 to Rq

(3) f(0)	XEQ 'CB'	→	f(0)
f(1)	R/S	→	f(1)
.			.
.			.
.			.
f(INT N/2)	R/S	→	f(INT N/2)
f(1+INT n/2)	R/S	→	f(1+INT N/2)
.			.
.			.
.			.
f(N-1)	R/S	→	-f(N-1)
f(N)	R/S	→	$c_4C_4(0)$

(4a) If $c_4C_4(0)$ is small enough to drop the quartic term, key

User A → α_0 (F 0 set)

SST, SST, SST → α_1, α_2, α_3

(4b) If quartic term is to be retained key

R/S → α_0

SST, SST, SST, SST → α_1, α_2, α_3, α_4

(5) To tabulate closeness of fit to data, turn off calculator, attach printer in MAN mode, turn on calculator and printer, and key User D to obtain print out of f and $(\hat{f} - f)$ for each data point where $\hat{f}$ is value calculated from polynominal equation. After the deviations are printed,

$s_y = \sqrt{\dfrac{\Sigma(\hat{f}-f)^2}{N-1}}$ and the average deviation, $\Sigma|(\hat{f}-f)|/(N+1)$ are printed.

(6) x_i ↑ I User E → a_0
SST, SST, SST, (SST) → a_1, a_2, a_3, (a_4)

(7) $\bar{x}$ User B $\hat{f}(\bar{x})$

(8) x User C $\hat{f}(x)$

If it is desired to check any constants, α_0 to α_4 are stored in registers R 11-15 and a_0 to a_4 are stored in R 16-20. All of the data points are stored in order from f(0) to f(N). The number of the last f register, which contains f(N) is given by 1000 (decimal portion of the f Index in R5).

If it is desired to fit another set of data with the same number of data points, it is not necessary to repeat steps (1), (2a) or (2b); one can start inserting the data with step (3). If it is desired to have dimensionless values of $-(G-H^\circ_{298})/RT$ using a_0/R to a_4/R, store R in register 8, turn printer on, and key User 'F'.

Test:

$-(G^\circ-H^\circ_{298})/T$ values for C(graphite) from JANAF Tables (3/31/78) were fit between 300 and 1300K.

(2a) 11 XEQ CH, SIZE? = > 52, F1 set, SIZE 052; (2b) R/S 21.040;

(3) 1.372 XEQ CB 1.3720, 1.462 R/S 1.4620, 1.657 R/S, 1.903 R/S, 2.174 R/S, 2.457 R/S 2.4570, 2.743 R/S -2.7430, 3.026 R/S -3.026, 3.306 R/S, 3.579 R/S -3.5790, 3.845 R/S 0.016;

(4b) R/S 1.368 SST 4.94×10^{-2} SST 5.87×10^{-2} SST -6.09×10^{-3} SST 2.21×10^{-4};

(5) User D

1.372	-0.004
1.462	0.003
1.657	-0.001
1.903	-0.005
2.174	-0.003
2.457	0.001
2.743	0.004
3.026	0.004
3.306	-0.002
3.579	-0.005
3.845	0.003
0.005	***
0.004	***

(6) 300 ↑ 10^2 E 1.9304, SST -4.9088×10^{-3}, SST 1.254×10^{-5}, SST -8.7446×10^{-9}, SST 2.212×10^{-12}; 1.98719 ST/16, ST/17, ST/18, ST/19, ST/20, RCL 16 to 20 or 1.98719 STO8 F

R16=	971.415-03
R17=	-2.47022-03
R18=	6.31089-06
R19=	-4.40047-09
R20=	1.11290-12

$-(G-H^o_{298})/RT = 0.9714 - 2.4702x10^{-3}T + 6.311x10^{-6}T^2 - 4.4005x10^{-9}T^3 +$
$1.113x10^{-12}T^4$, 300-1300K, deviations range from

(7) 2 B $-(G-H^o_{298})/T = 1.6564$ compared with tabulated 1.657 cal

(8) 10^3 C $-(G^o-H^o_{298})/RT = 1.5245$, R/S 3.0295 compared to tabulated $-(G^o-H^o_{298})/T =$
3.026

Register Use

R 00	1	2	3	4	5	6	7	8	9	10
N	$\frac{N}{2}$	Initial C_n Index $21+\frac{q}{1000}$	C_n index $20+\frac{q}{1000}$ to $q+1+\frac{q}{1000}$	q	q+1	n index 1 to 3	$\bar{x}$ index 0 to INT $\frac{N}{2}$			
	m index 0 to 3, 0 to 2, 0 to 1, 0		q+1 to 20	(f,C_n) index	f index	$(f,C_0) = \Sigma f(x)$	(f,C_1)	(f,C_2)	(f,C_3)	(f,C_4)
			α index 15 to 11							
	I		a index 20 to 16	n index 4 to 1						

11	12	13	14	15
$\frac{(f,C_4)}{(C_4,C_4)}$				
$\alpha_0 = \Sigma\frac{(f,C_n)}{(C_n,C_n)}$	α_1	α_2	α_3	$[\alpha_4]$

16	17	18	19	20
		(C_{43})	(C_{42})	$\frac{(f,C_4)}{(C_4,C_4)}$ or $C_{4,1}$
	C_{32}	C_{33}	$C_{31}-C_{32}$	$(C_{41}-C_{42})$
	C_{22}	$\frac{(f,C_2)}{(C_2,C_2)}$	$\frac{(f,C_3)}{(C_3,C_3)}$	$(C_{41}-C_{42}+2C_{43})$ as $B_{31}=2$
				(C_{44})
a_0	a_1	a_2	a_3	a_4

21	22	23	24	25	---- q = 20+2(N-1) for N odd or 20+2N for N even
$C_1(1)$	$C_2(1)$	$C_3(1)$	$C_4(1)$	$C_1(2)$	--- $C_4(\mathrm{INT}\ \frac{N}{2})$
q+1	q+2	---- q+N+1 = 19+3N for N odd, 21+3N for N even			
f(0)	f(1)	---- f(N)			

SIZE = 20+3N odd
22+3N even

One may not have values of f(x) to be inserted in step (3) and it may be necessary to calculate f(x,y) from values of x and y. For example, one might wish to express $-\Delta(G_T^o-H_{298}^o)/RT$ as a polynomial in T given values of ΔG_T^o at even T intervals: $-\Delta(G_T^o-H_{298}^o)/RT = (\Delta H_{298}^o-\Delta G_T^o)/RT$. One would make the following modifications to program CB. RCL Z of step 6 would be replaced by the two steps R↑ R↑ and the following six steps would be added to the beginning of LBL 10: RCL 55 + RCL56 / x⇄y / ΔH_{298}^o would be stored in R55 and R would be stored in R56. SIZE would be set at 57. With these program modifications, T↑ $-\Delta G_T^o$ would be inserted in place of each f(x) of step (3). Of course ΔH and ΔG have to be in the same units and R has to be in corresponding units. The values of ΔG^o are not stored, but the derived values of $-\Delta(G^o-H_{298}^o)/RT$ are stored.

```
 01♦LBL "CH"
1  -  STO 00  2  /
STO 01  RCL 01  INT
X≠Y?  GTO 04  SF 01
GTO 05

 14♦LBL 04
CF 01

 16♦LBL 05
4  *  20  +  STO 04
1 E03  /  21  +  STO 02
1  -  STO 03  RCL 04
RCL 00  +  2  +  FIX 0
"SIZE?=>0"  ARCL X
PROMPT  0  FIX 3  STO 07

 42♦LBL 00
ISG 03  GTO 03  RCL 02
RTN

 47♦LBL 03
1  STO 06  ST+ 07
RCL 07  RCL 01  /  CHS
1  +  STO IND 03

 58♦LBL 01
2  *  RCL 01  RCL 07  -
*  RCL 06  2  *  1  +  *
X<>Y  RCL 06  *  RCL 00
RCL 06  +  1  +  *  -
RCL 00  RCL 06  -  /
RCL 06  1  +  /  ISG 03
STO IND 03  4  RCL 06  1
+  X=Y?  GTO 00  STO 06
1  ST- 03  R↑
RCL IND 03  X<>Y  ISG 03
GTO 01


105♦LBL B
RCL 02  STO 03

108♦LBL 02
FIX 0  RCL 03  PSE
RCL IND 03  FIX 2  PSE
ISG 03  GTO 02  RTN

118♦LBL C
RCL 02  PRREGX  END
```

121 steps
167 bytes

```
 01♦LBL "CB"
6.1  STO 04  RCL 02
STO 03  RCL Z  XEQ 11
STO 06  STO 07  STO 08
STO 09  STO 10  RTN
XEQ 10

 15♦LBL 01
ENTER↑  ENTER↑  ENTER↑
ST+ IND 04  ISG 04
XEQ 02  XEQ 02  XEQ 02
XEQ 02  5  ST- 04  R↑
RTN  XEQ 10  GTO 01

 31♦LBL 02
RDN  RCL IND 03  *
ST+ IND 04  ISG 04
ISG 03  RTN  RDN  RCL 03
INT  STO 03  RDN  DSE 03
FS? 01  XEQ 04  STOP
XEQ 10  GTO 05

 50♦LBL 04
4  ST- 03  RDN  RTN

 55♦LBL 05
ENTER↑  ENTER↑  XEQ 09
ST+ 10  RDN  XEQ 06  RDN
XEQ 09  ST+ 09  RDN
XEQ 09  ST+ 08  RDN
XEQ 09  ST+ 07  R↑  STOP
XEQ 10  GTO 05

 75♦LBL 09
RCL IND 03  DSE 03  *
RTN

 80♦LBL 06
STO T  STO Y  CHS  STO Z
RDN  ST+ 06  RTN

 88♦LBL A
STO 20  0  STO 11
STO 12  STO 13  STO 14
SF 08  GTO 07

 97♦LBL 11
RCL 02  FRC  1 E3  *  1
+  STO 05  RCL 00  +
1 E3  /  ST+ 05  RDN
```

```
111♦LBL 10
STO IND 05  ISG 05  RTN
XEQ 06  ST+ 10  RDN
ST+ 09  RDN  ST+ 03  RDN
ST+ 07

123♦LBL 17
0  STO 01  4  STO 04
XEQ 13  RCL 10  *
STO 11  STOP  CF 00  1
XEQ 14  STO 20  1
STO 01  RDN  XEQ 14
STO 19  ST- 20  2
STO 01  RDN  XEQ 14
STO 18  2  *  ST+ 20  3
STO 01  RCL 18  XEQ 14
RCL 20  X<>Y  STO 20  6
*  -  RCL 11  *  STO 12
RCL 19  RCL 18  3  *  -
RCL 20  11  *  +  RCL 11
*  STO 13  RCL 18
RCL 20  6  *  -  RCL 11
*  STO 14  RCL 20
RCL 11  *  STO 15

188♦LBL 07
0  STO 01  3  STO 04  1
XEQ 14  STO 19  1
STO 01  RDN  XEQ 14
ST- 19  STO 17  2
STO 01  RDN  XEQ 14
STO 18  2  *  RCL 19  +
XEQ 13  RCL 09  *
STO 19  ST+ 11  *
ST+ 12  RCL 17  RCL 18
3  *  -  RCL 19  *
ST+ 13  RCL 18  RCL 19
*  ST+ 14  DSE 04  0
STO 01  1  XEQ 14
STO 19  1  STO 01  RDN
XEQ 14  STO 17  ST- 19
XEQ 13  RCL 08  *
ST+ 11  STO 18  RCL 19
*  ST+ 12  RCL 17
RCL 18  *  ST+ 13
DSE 04  0  STO 01  1
XEQ 14  XEQ 13  RCL 07
*  ST+ 11  *  ST+ 12
RCL 06  RCL 00  1  +  /
ST+ 11  RCL 11  STOP
RCL 12  RCL 13  RCL 14
RCL 15

277♦LBL 14
RCL 01  1  +  X↑2  LASTX
RCL 04  +  /  1/X
RCL 01  RCL 04  -  *
RCL 00  RCL 01  -  /  *
RTN

297♦LBL 13
RCL 00  FACT  X↑2
RCL 00  RCL 04  -  FACT
/  RCL 00  RCL 04  +  1
+  FACT  /  RCL 04  2  *
1  +  *  RTN

320♦LBL E
STO 01  RDN  ENTER↑
ENTER↑  ENTER↑  FC? 00
XEQ 15  FS? 00  XEQ 00
RCL 14  RCL 01  3  Y↑X
/  ST+ 19  *  3  *
ST- 18  *  ST+ 17  *  3
/  ST- 16  RDN  RCL 13
RCL 01  X↑2  /  ST+ 18
*  2  *  ST- 17  *  2  /
ST+ 16  RDN  RCL 12
RCL 01  /  ST+ 17  *
ST- 16  RCL 11  ST+ 16
RCL 16  RTN  RCL 17
RCL 18  RCL 19  RCL 20

375♦LBL 15
RCL 15  RCL 01  4  Y↑X
/  STO 20  *  CHS  4  *
STO 19  *  CHS  1.5  *
STO 18  *  CHS  1.5  /
STO 17  *  CHS  4  /
STO 16  RDN  RTN

404♦LBL 00
0  STO 16  STO 17
STO 18  STO 19  RDN  RTN

412♦LBL 16
RCL IND 05  ACX  3
SKPCHR  X<> T  RDN  -
ACX  ADV  ABS  RTN

424♦LBL D
FIX 3  RCL 00  1  +
ST- 05  RCL 11  XEQ 16
STO 06  X↑2  STO 07  1
ST+ 05

437♦LBL 08
15  XEQ 03  XEQ 16
ST+ 06  X↑2  ST+ 07  R↑
1  +  ISG 05  GTO 08
RCL 00  1  +  ST/ 06  2
-  ST/ 07  RCL 07  SQRT
PRX  RCL 06  PRX  RTN

462♦LBL B
15  GTO 03

465♦LBL C
20

467♦LBL 03
STO 03  FS? 00  DSE 03
RDN  ENTER↑  ENTER↑
ENTER↑  RCL IND 03  *
DSE 03  XEQ 12  XEQ 12
FC? 00  XEQ 12
RCL IND 03  +  RTN
RCL 03  *  RTN

489♦LBL 12
RCL IND 03  +  *  DSE 03
RTN

494♦LBL F
RCL 08  ST/ 16  ST/ 17
ST/ 18  ST/ 19  ST/ 20
ENG 5  16.02  FS? 00
16.019  PRREGX  END
```

506 steps

759 bytes

CHAPTER III

Least-Square Fitting of Data to an Analytical Function

Least-square fitting of data to any equation $y = f(x)$ is not a routine process but requires careful consideration of the variations of errors in y as a function of x.[4,5,6] For example, if it were desired to obtain the values of a and b in the expression $y = ax^2 + bx^3$ that best represent a set of data, one could least-square a variety of functions of y. The use of the unweighted function would tend to heavily weight values of y at large x. As just one alternative example, one could least-square $y/x^2 = a + bx$ and obtain, in general, quite different values of a and b that would correspond to more heavy weighting of values of y at low x than for the previous procedure.

One should carefully consider the magnitude of errors in y as a function of x before selecting the appropriate procedure. One should apply appropriate weighting to off-set any bias of the least-square procedure as well as to attempt to correct for systematic errors.[6]

A set of x,y values may be fit by least-squaring procedures to a variety of equations. Unless the data are unusually accurate, or have been smoothed to fit an equation closely, it is rare that more than three parameters are justified. The four equations that are fit by the least-square program given here are $f(y) = a + bf(x')$, $f(y) = af_1(x') + bf_2(x')$, $f(y) = a + bf_1(x') + cf_2(x')$, and $f(y) = af_1(x') + bf_2(x') + cf_3(x')$ which will be identified in the program as ab1, ab2, abc2, and abc_3. $f(y)$ can be any function of y or of x and y and f_1, f_2, and f_3 can be any three different functions of x or x', where x' which is a function of x such as $f(T') = T-298$, $T-1000$, $2890-T$, $T/298$, etc. $f^{-1}(y)$ must also be specified to convert values of $f(y)$ to values of y.

As pointed out above, the least-square process can not be an automatic procedure. Built-in weighting bias can distort the fitting depending upon the way in which y varies with x. One can offset the bias as illustrated above by fitting $y/x^2 = a + bx$ instead of $y = ax^2 + bx^3$ by switching from program ab2 to ab1. The least-square program also allows specific weighting factors to be applied to each specific value.

In applying the least-square program, one must first make a decision about which of the four equations will be used. Then one must decide whether individual weighting will be used. If the values of x are spaced at even intervals, the insertion of the data can be simplified by storing the value of I, the interval between x values.

All data are stored and can be retrieved to be fit to any other equations, if desired. Once the constants a and b or a,b, and c have been fixed, the program will provide calculated values of y, $\hat{y}$, for any value of x in the range that was fit. If it is desired to examine the nature of the fit, insertion of the HP41c printer will provide a print-out of $\hat{y}$-y values for all n values of the original data, the standard deviation $\sqrt{(\hat{y}-y)^2/(n-2)}$ and $(\hat{y}-y)/n$, the average deviation. For accurate statical evaluation, the standard deviation expression should be modified by replacing the 2 in n-2 by larger values depending upon the degree of the equation being fit.

To illustrate the selection of $f(y)$, $f^{-1}(y)$, x', $f_1(x')$, etc., some specific examples will be given. High temperature heat capacities are often obtained by Drop Calorimetry. Drop Calorimetry yields values of H_T-H_i for various values of T, where i refers to the reference temperature which may be 273 or 298K or some other calorimeter temperature. It is desired to obtain a C_P equation which will join the accurately known C_P at temperature T_i from low temperature. Shomate has proposed an equation which has been found useful for C_P evaluation.[7,8] For $C_P = a + bT + c/T^2 + dT^2$, $(H_T-H_i)/\Theta^2 - C_{P,i}/\Theta = a_0 + a_1\Theta + a_{-1}(\Theta+T_i)^{-1}$ where $\Theta = T-T_i$, $d = 3a_1$, $c = -T_i^2a_{-1}$, $b = 2a_0 - dT_i$ and $a = C_{P,i} - bT_i - cT_i^{-2} - dT_i^2$. For use in the least-square program $y = H_T-H_i$, $x = T$, $x' = T-T_i = \Theta$, $f(y) = y/\Theta^2 - C_{P,i}/\Theta$, $f^{-1}(y) = \Theta^2 f(y) + \Theta C_{P,i}$, and $f_2(x') = (\Theta + T_i)^{-1}$ in program abc2.

This procedure joins C_P smoothly to the low temperature values, but the derivative may be discontinuous. To ensure a smooth joining, one would use $f(y) = y/\Theta - C_{P,i} - \frac{1}{2}\Theta(dC_{P,i}/dT)$ and $f^{-1}(y) = \Theta(f(y) + C_{P,i}) + \frac{1}{2}\Theta^2(dC_{P,i}/dT)$ with $f_1(x') = (\Theta/T_i^2)(1/T_i - 1/T) = \Theta^2/T_i^3T$ and $f_2(x') = (1/3)\Theta^2$ in program ab2 which will yield the constants c and d of the C_P equation. The other two constants are given by $b = dC_{P,i}/dT + 2(c/T_i^3 - dT_i)$ and $a = C_{P,i} - bT_i - c/T_i^2 - dT_i^2$.

There may be no accurate low temperature heat capacity data and the high temperature data may not be accurate enough to warrant four parameters. However, the temperature T* at which dC_p/dT reaches a minimum is clearly indicated by the heat capacity data. Use of T* reduces the parameters to three with $y = C_p$, $x' = T$, $f_1(x) = (T + T_i)/2$ and $f_2(x) = 1/T^2 - 3T^2/(T^*)^4$ with $d = -3c/(T^*)^4$. a, b, and c are given by program abc2.

A similar treatment for enthalpy with $y = (H_T - H_i)$, $f(y) = y/\Theta$, $f^{-1}(y) = \Theta f(y)$, $x' = x$ ' T, $f_1(x) = \frac{1}{2}(T+T_i)$, and $f_2(x) = 1/TT_i - (T^2+TT_i+T_i^2)/(T^*)^4$ yields with program abc2 values of a, b, and c. $d = -3c/(T^*)^4$.

The example of the regular solution partial molal equation $\bar{Y}_1 = ax_2^2 + bx_2^3$ with the choice of $f(y) = \bar{Y}_1$, $\bar{Y}_1/x_2$, and $\bar{Y}_1/x_2^2$ with appropriate changes in f_1 and f_2 has been discussed above. A related equation for the integral quantity $Y_1/x_1x_2 = a + \frac{1}{2}b + \frac{1}{2}bx_2$ can be treated with $f(y) = y/(1-x_2)x_2$, $f^{-1}(y) = (1-x_2)x_2f(y)$, and $f(x) = \frac{1}{2}x_2$. The term designated as b by program ab1 corresponds to the b term of the regular solution equation. The term designated as a by program ab1 is equal to $\frac{1}{2}$b plus the regular solution a term.

When regular solution theory is applied to calculation of solidus and liquidus curves of phase diagrams, an explicit equation for the boundaries can not be obtained when there is appreciable solid and liquid solubility, although accurate values can be calculated by successive approximations.[(2)] The calculated values can be expressed analytically in terms of an approximate equation of the form that would apply if solubilities were small plus a deviation function. A least square fit using program abc2 can provide an accurate representation of the solidus and liquidus boundaries.[(2)]

Directions

If program is already in or after insertion of cards, indicate by 1a, 1b, 1c, or 1d which equation will be used to fit data. ' ' indicates ALPHA mode.

(1a) $f(y) = a + bf(x')$, key XEQ ' a b 1', which sets F1.

(1b) $f(y) = af_1(x') + bf_2(x')$, key XEQ ' a b 2', which sets F2.

(1c) $f(y) = a + bf_1(x') + cf_2(x')$, key XEQ ' a b c 2', which sets F3.

(1d) $f(y) = af_1(x') + bf_2(x') + cf_3(x')$, key XEQ ' a b c 3', no flag set.

For all four program initiations, the calculator will prompt w? SF0 I? STO 00. If all the data are to be given equal weighting, no response to the first question is needed. If w≠1 for any data, key SF 00. If the values of x are at regular intervals of I, store I in R00; otherwise no response is needed.

If SIZE is not sufficient, XEQ 'SIZE' 22+2n, where n is the number of data sets to be entered. If f(y) and f(x) have not already been inserted for the desired equation, step (2) is carried out.

			Display
(2a)	key PRGM	→	LBL1, key in f(y).
(2b)	SST SST	→	LBL2, key in $f^{-1}(y)$.
(2c)	SST SST SST	→	STO 06, key in x'=f'(x). If x'=x, nothing is keyed in.
(2d)	SST	→	STO 05, key in $f_1(x')$. If f(x')=x', nothing is done.
(2e)	SST SST SST	→	RCL 05 — for ab1, key 0 X. — for other programs, key $f_2(x')$.
(2f)	SST SST SST	→	RCL 05 — for ab3, key $f_3(x')$ PRGM. — otherwise, key 0 X PRGM.

The above instructions assume no f(y) or f(x) steps left over from previous calculations; if there are, they must be deleted if not consistent with the desired functions. If there are no plans to use abc3 in a series of calculations, step (2f) can be eliminated by leaving 0 X in LBL5 and completing step (2c) with PRGM. If f(x')=x, steps (2c) and (2d) can be bypassed after step (2b) by PRGM GTO4 PRGM SST followed by step (2c). Other simplifications of step (2) are possible using GTO and BST.

For each program, there are four variants for inserting data.

(3a)	no I, w=1	key	$x_1 \uparrow y_1$	User $E \rightarrow x_1$
			$x_2 \uparrow y_2$	$E \rightarrow x_2$
			$\vdots \quad \vdots$	$\vdots \quad \vdots$
			$x_n \uparrow y_n$	$E \rightarrow x_n$
(3b)	I STO 00, w=1	key	$x_1 \uparrow y_1$	User $E \rightarrow x_2$
			y_2	$E \rightarrow x_3$
			$\vdots$	$\vdots \quad \vdots$
			y_n	$E \rightarrow x_{n+1}$
(3c)	no I, w≠1 SF00	key	$x_1 \uparrow y_1 \uparrow w_1$	User $E \rightarrow x_1$
			$x_2 \uparrow y_2 \uparrow w_2$	$E \rightarrow x_2$
			$\vdots \quad \vdots \quad \vdots$	$\vdots \quad \vdots$
			$x_n \uparrow y_n \uparrow w_n$	$E \rightarrow x_n$
(3d)	I STO 00, w≠1 SF 00	key	$x_1 \uparrow y_1 \uparrow w_1$	User $E \rightarrow x_2$
			$y_2 \uparrow w_2$	$E \rightarrow x_3$
			$\vdots \quad \vdots$	$\vdots \quad \vdots$
			$y_n \uparrow w_n$	$E \rightarrow x_{n+1}$

(4) After all data have been entered,
RS → a, SST → b, and SST → c for abc2 and abc3.
Calculated values of y, $\hat{y}$, can be calculated for any x using step 5a.
If values are desired at even intervals of x, step 5b can be used.

(5a) x User C → y

(5b) If I in R00, x_1 User C → y_1, R/S → y_2, R/S....

(6) The closeness of the fit between the calculated y values and the unweighted original data can be checked by turning off the calculator, attaching the printer, turning both on with the printer in MAN mode, and keying n XEQ 10. The printer will print y_1, $\hat{y}_1 - y_1$; y_2, $\hat{y}_2 - y_2$ ... y_n, $\hat{y}_n - y_n$ followed by $\sqrt{\Sigma(\hat{y}-y)^2/(n-2)}$ and $(\Sigma|\hat{y}-y|)/n$. (If the printer is still attached from a previous step 6 when step 1 is carried out, the calculator will stop after display of w?SF0. R/S will complete the display of I?STO 00. SST will then put calculator in position for

insertion of f(y).)

(7) The x_i, y_i values were stored in step (3) and can be retrieved to fit to another equation or other functions. Repeat step (1) to indicate which equation and insert desired functions. Then n User A will retrieve x_i, y_i values and make an unweighted least square fit in place of step (3). Follow with step (4) to obtain a, b, and c values. If it is desired to use weighting in the new fit or change the weights applied in a previous calculation, subroutine LBL6 can be modified by inserting RTN before XEQ E and before GTO E. Step (1) is followed by SF 00 and n User A as indicated above, but the calculation will stop to display each y_i. Then key in w_i followed by R/S. After the last value has been keyed in, follow as usual with step (4).

---- Note 1: Additional data can be inserted by step (2) after steps 4, 5, 6, or 7 if the appropriate flag is set for the equation being used, and SIZE is adequate or is increased.

---- Note 2: Steps 5, 6 and 7 can be repeated in any order.

---- Note 3: The closeness of fit obtained in step (6) can be compared with the fit using weighting by inserting RTN X in LBL 14 between PRX and ABS. Σw_i which can be obtained from R16 for all programs except abc3 is used in place of n in initiating step (6). The calculation will stop after the display of each $\hat{y}_i - y_i$. Key in w_i followed by R/S. After the last value, there will be two additional printouts of which the last will be $\sum_i w_i|\hat{y}_i - y_i|/\Sigma w_i$ which can be compared to $\sum_i |\hat{y}_i - y_i|/n$ given by step 6 as normally carried out.

---- Note 4: If f(y) and f(x) used in the previous calculation require three or more steps, the delete function can be used to remove them and add the functions needed for the current calculation. If the previous calculation was the fitting of enthalpy data to match $C_{P,i}$ and $(dC_P/dT)_i$ with ab2 as described in the introductory text of Chapter III, f(y) took 14 steps, $f^{-1}(y)$ 8 steps, x' 2 steps, f_1 9 steps, and f_2 3 steps. Step (2) would be carried out as follows:

□GT01 PRGM SST XEQ 'DEL' 014, key in new f(y), SST SST SST XEQ 'DEL' 008, key in new $f^{-1}(y)$, SST SST SST SST SST ← ←, key in new x', SST SST XEQ 'DEL' 009, key in new f_1, SST SST SST XEQ 'DEL' 003, key in new f_2, PRGM if no new f_3.

As mentioned in the step (2) instructions, one can use PRGM □GTO5, for the example of deletion of f_2, PRGM □BST □BST ← ← ← key in new f_2 PRGM. One could reduce the number of keys required by three if the step number after inserting f_1, e.g. 40, is noted. Key □GTO .046 ← ← ←, key in new f_2, PRGM.

<u>TESTS</u> A sample calculation is carried out for each of the four programs which can serve as a check if the program is operating properly. The appendix to Chapter III gives insertions into the program for the functions and sample calculations for the fitting of Drop Calorimetry data as discussed in the introductory text of Chapter III.

<u>ab1 Test</u>: $\ln y = a + bx^{-1}$, n=4, I=100.

(1a) XEQ '□a□b□1' → F1, EEX 2 STO 00, 'SIZE' 030

(2) With no entries from a previous calculation to be removed,

PRGM LN SST SST □e^x SST SST SST SST 1/x
SST SST SST 0 X SST SST SST 0 X PRGM

(3b) 1300 ↑ 0.0147 User E → 1400, 0.0263 E → 1500, 0.045 E → 1600, 0.0696 E → 1700

(4) R/S a=4.108, SST b=-10 830

(5b) 1300 C 0.01465, R/S 0.02657, R/S 0.0445, R/S 0.06988

(6) OFF, Printer in, MAN, ON ON 4 XEQ 10

```
0.01470   -0.00005
0.02630    0.00027
0.04500   -0.00050
0.06960    0.00028

   0.00045    ***
   0.00027    ***
```

(7) Retrieval for weighted fit:

XEQ '□a□b□1' → F1, □SF00, EEX 2 STO 00; □GTO 6 PRGM SST □RTN □GTO.508 □RTN PRGM; 4 User A 0.0147, 2 R/S 0.0263, 4 R/S 0.0450, 1.5 R/S 0.0696, 3 R/S 1700; R/S a = 4.106, b = -10 831

□GTO 6 PRGM SST SST ← □GTO.508 ← PRGM, to delete RTN twice.

<u>ab2 Test</u>: $y = ax^2 + bx^3$, n = 9, I = 0.1, values to be weighted

(1b) XEQ '□a□b□2' → F2, □SF00 'SIZE' 040, 0.1 STO 00

(2) PRGM SST ← □GTO.038 ← □GTO.042 ← □x^2 □GTO.047 ← ← 3 □y^x PRGM

(3d) 0.1 ↑ 4.001 ↑ 10 User E 0.2, 23.998 ↑ 9 E 0.3, 72.003 ↑ 8 E 0.4, 159.996 ↑ 7 E 0.5, 300.005 ↑ 5 E 0.6, 503.994 ↑ 4 E 0.7, 784.007 ↑ 3 E 0.8, 1151.992 ↑ 2 E 0.9, 1620.009 ↑ 1.5 E 1.0

(4) R/S a = 199.9906, SST **b** = 2000.015

(5b) 0.1 C 4.000, R/S 24.000, R/S 72.000, 0.9 C 1620.003

(6) Printer ON, 9 XEQ 10,

4.001	-0.001
23.998	0.002
72.003	-0.003
159.996	0.003
300.005	-0.005
503.994	0.006
784.007	-0.006
1,151.992	0.010
1,620.009	-0.006
0.006	***
0.005	***

abc2 Test: y = a + blnx + c/x, n=3, weighted fit

(1c) XEQ '□a□b□c□2' → F3, □SF00, 'SIZE' 028

(2) □GTO 4 PGRM □BST □BST ← LN □GTO.047 ← ← 1/x PRGM

(3c) 1 ↑ 20 ↑ 2 E 1, 10 ↑ 15.605 ↑ 1.5 E 10, EEX 2 ↑ 19.31 ↑ 1 E 100

(4) R/S a = 10.00, SST b = 1.99993, SST c = 10.00

(5) 1 C 20.0, 10 C 15.605, EEX 2 C 19.31

abc3 Test: $1/y = a(3000 - x) + b(3000 - x)^2 + c(3000 - x)^3$, n=5, I=100

(1d) XEQ '□a□b□c□3', SF00, EEX 2 STO 00, 'SIZE' 032

(2) PRGM 1/x SST SST 1/x SST SST SST 3 EEX 3 - CHS SST SST ← GTO.050 ← □x^2 □GTO.055 ← ← 3 □y^x PRGM

(3d) 1800 ↑ 2.2894×10^{-4} ↑ 1 E 1900, 2.7465×10^{-4} ↑ 2 E 2000, 3.3333×10^{-4} ↑ 3 E 2100, 4.1×10^{-4} ↑ 4 E 2200, 5.1234×10^{-4} ↑ 5 E 2300

(4) R/S a = 0.99508, SST b = 1.00947×10^{-3}, c = 9.955×10^{-7}

(5) 1800 C 2.2894×10^{-4}, R/S 2.74645×10^{-4}, R/S 3.33328×10^{-4}, R/S 4.1001×10^{-4}, R/S 5.1234×10^{-4}

The keying of y could have been simplified by changing f(y) to $10^4/y$ and keying in 10^4y.

```
01•LBL "ab1"
SF 01  GTO 00

04•LBL "ab2"
SF 02  GTO 00

07•LBL "abc2"
SF 03  GTO 00

10•LBL "abc3"
11•LBL 00
CF 00  22.1  STO 20  CLΣ
0  STO 00  STO 01
STO 03  STO 04  FS? 01
GTO 13  STO 17  STO 18
FS? 02  GTO 13  STO 19

28•LBL 13
"N? SF 0"  AVIEW  PSE
"I? ST 00"  AVIEW  RTN

35•LBL 01
RTN

37•LBL 02
RTN

39•LBL 03
STO 06  STO 05  RTN

43•LBL 04
RCL 05  RTN

46•LBL 05
RCL 05  RTN

49•LBL E
FS? 00  GTO 07  1
STO 08  RDN  GTO 08

56•LBL 07
ST+ 16  SQRT  STO 08  1
ST- 16  RDN  RDN

64•LBL 08
STO IND 20  ISG 20
XEQ 01  RCL 08  *
STO 10  X<>Y  STO IND 20
ISG 20  XEQ 02  RCL 08
*  STO 09  FS? 01
GTO 09  XEQ 04  RCL 08
*  STO 07  FS? 02
GTO 11  FS? 03  GTO 12
GTO 15

89•LBL 09
RCL 08  1  -  *  ST+ 11
LASTX  RCL 10  *  ST+ 13
RCL 10  RCL 09  Σ+
RCL 06  RCL 00  +  RTN
RCL 15  RCL 11  RCL 13
*  RCL 16  /  -  RCL 12
RCL 11  X↑2  RCL 16  /
-  /  STO 02  MEAN
RCL 02  *  -  STO 01
CF 01  RTN  RCL 02

129•LBL 11
RCL 10  *  ST+ 18
RCL 09  RCL 10  *
ST+ 17  RCL 07  RCL 09
Σ+  RCL 06  RCL 00  +
RTN  RCL 12  RCL 18  *
RCL 17  RCL 15  *  -
RCL 12  RCL 14  *
RCL 15  X↑2  -  /
STO 03  RCL 15  *
RCL 17  -  CHS  RCL 12
/  STO 02  CF 02  RTN
RCL 03

170•LBL 12
RCL 08  1  -  STO 03  *
ST+ 13  RCL 03  RCL 09
*  ST+ 11  RCL 10
RCL 08  *  ST+ 19
RCL 07  XEQ 11  RTN
RCL 19  STO 10  RCL 11
X↑2  RCL 16  ST/ 10  /
RCL 12  -  RCL 11
RCL 13  RCL 16  /  *
RCL 15  -  STO 06  X<>Y
STO 03  RCL 13  X↑2
RCL 16  /  RCL 14  -
STO 05  RCL 17  RCL 18
RCL 10  RCL 13  *  X<>Y
-  STO 08  RCL 10
RCL 11  *  R↑  -  STO 07
RCL 03  RCL 05  *
RCL 06  X↑2  -  RCL 07
RCL 05  *  RCL 06
RCL 08  *  -  X<>Y  /
STO 02  RCL 03  *
RCL 07  -  CHS  RCL 06
/  STO 03  RCL 13  *
RCL 11  RCL 02  *  +
RCL 16  /  RCL 10  -
CHS  STO 01  CF 03  RTN
RCL 02  RCL 03

268•LBL 15
RCL 10  *  ST+ 12
RCL 09  X↑2  ST+ 14
RCL 07  X↑2  ST+ 17
RCL 09  RCL 07  *
ST+ 15  RCL 10  RCL 09
*  ST+ 11  XEQ 05
RCL 08  *  ENTER↑
ENTER↑  ENTER↑  X↑2
ST+ 19  RDN  RCL 10  *
ST+ 13  RDN  RCL 09  *
RCL 08  X↑2  -  1  +
ST+ 16  RDN  RCL 07  *
ST+ 18  RCL 06  RCL 00
+  RTN  RCL 17  RCL 19
*  RCL 18  X↑2  -
STO 04  RCL 15  RCL 19
*  RCL 16  RCL 18  *  -
STO 05  RCL 15  RCL 18
*  RCL 16  RCL 17  *  -
STO 06  RCL 16  *
RCL 14  RCL 04  *  +
RCL 15  RCL 05  *  -
STO 07  RCL 12  RCL 18
*  RCL 13  RCL 17  *  -
STO 08  RCL 16  *
RCL 12  RCL 19  *
RCL 13  RCL 18  *  -
STO 09  RCL 15  *  -
RCL 04  RCL 11  *  +
RCL 07  /  STO 02
RCL 13  RCL 15  *
RCL 12  RCL 16  *  -
STO 10  RCL 16  *
RCL 05  RCL 11  *  -
RCL 09  RCL 14  *  +
RCL 07  /  STO 03
RCL 06  RCL 11  *
RCL 10  RCL 15  *  -
RCL 08  RCL 14  *  -
RCL 07  /  STO 04
RCL 02  RTN  RCL 03
RCL 04
```

```
416♦LBL C
XEQ 03  RCL 02  *
XEQ 04  RCL 03  *  +
XEQ 05  RCL 04  *  +
RCL 01  +  XEQ 02  RTN
RCL 00  RCL 06  +  GTO C

436♦LBL 10
STO 07  2  *  22.02  +
1 E3  /  22  +  STO 20
1  +  STO 21  0  STO 09
STO 10

453♦LBL 14
RCL IND 21  XEQ C
RCL IND 20  ISG 20  ACX
3  SKPCHR  RDN  -  ACX
ADV  ABS  ST+ 10  X↑2
ST+ 09  ISG 21  GTO 14
RCL 07  ST/ 10  2  -
ST/ 09  RCL 09  SQRT
ADV  PRX  RCL 10  PRX
RTN

483♦LBL A
2  *  19  +  1 E3  /  22
+  STO 21

493♦LBL 06
XEQ 16  XEQ E  ISG 21
GTO 06  .1  ST+ 21
XEQ 16  GTO E

502♦LBL 16
RCL IND 21  ISG 21
RCL IND 21  X<>Y  END
```

507 steps

713 bytes

All programs start with 0 in R00 which is replaced by I if x values are at regular intervals of I. R20 contains the index for storing y,x values. It starts with 22.1 and increments to 22.1+2n. R21 is used for a x index for LBL10 and LBL14 and for another y,x index used in LBLA, LBL6, and LBL16 which starts at 22 + 0.019 + 2n/1000. Registers 11-16 are cleared upon initiation.

R	1	2	3	4	5	6	7	8	9	10	11	12	13
ab1	0 a	b	0	0	x'	x	--- n	$\sqrt{w}$	$f\sqrt{w}$	$y\sqrt{w}$	Σfw	Σf^2w	Σyw
ab2	0	a	0 b	0	x'	x	$f_2\sqrt{w}$ n	$\sqrt{w}$	$f_1\sqrt{w}$	$y\sqrt{w}$	---	Σf_1^2w	---
abc2	0 a	b	$\sqrt{w}-1$ q c	0	x' r	x s	$f_2\sqrt{w}$ t n	$\sqrt{w}$ u	$f_1\sqrt{w}$	$y\sqrt{w}$ $\Sigma yw/\Sigma w$	Σf_1w	Σf_1^2w	Σf_2w
abc3	0	a	0 b	0 A c	x' B	x C	$f_2\sqrt{w}$ D n	$\sqrt{w}$ R	$f_1\sqrt{w}$ Q	$y\sqrt{w}$ S	Σyf_1w	Σyf_2w	Σyf_3w

R	14	15	16	17	18	19
ab1	---	Σyfw	Σw			
ab2	Σf_2^2w	Σf_1f_2w	---	0 Σyf_1w	0 Σyf_2w	
abc2	Σf_2^2w	Σf_1f_2w	Σw	0 Σyf_1w	0 Σyf_2w	0 Σyw
abc3	Σf_1^2w	Σf_1f_2w	Σf_1f_3w	0 Σf_2^2w	0 Σf_2f_3w	0 Σf_3^2w

y_1 is stored in R22, x_1 in R23, y_2 in R24, etc. until x_n is stored in R21+2n.

APPENDIX I (for Chapter I)

Prgm INTERP

2 Pt: $a_1 = (y_1-y_2)/(x_1-x_2)$, $a_0 = y_2 - a_1x_2$

3 Pt: $$a_1 = \left[\frac{(y_2-y_1)(x_2+x_3)}{x_2 - x_1} - \frac{(y_3-y_2)(x_1+x_2)}{x_3 - x_2}\right] \Big/ (x_3-x_1)$$

$$a_2 = \left(\frac{y_3-y_2}{x_3-x_2} - a_1\right) \Big/ (x_2+x_3)$$

$$a_0 = y_2 - a_1x_2 - a_2x_2^2$$

4 Pt: $$y_n^* = y_n - y_1$$

$$a_3 = (y_2^* - y_3^* + y_4^*/3)/2I^3$$

$$a_2 = (y_3^* - 2y_2^*)/2I^2 - 3a_3(I + x_1)$$

$$a_1 = y_2^*/I - a_2(I + 2x_1) - (I^2 + 3x_1I + 3x_1^2)a_3$$

$$a_0 = y_1 - a_1x_1 - a_2x_1^2 - a_3x_1^3$$

When values of y are not directly available and must be calculated, the program can easily be modified. Steps a to d for the calculation of $-\Delta(G^\circ-H^\circ_{298})/RT$ present one example of the ways in which f(x) values can be calculated and then fit to a power series. As another example, the use of values of ΔH°_{298} and ΔG°_T to obtain a two to four point fit for $-\Delta(G^\circ_T-H^\circ_{298})/RT$ will be illustrated. Key 0 ↑ ↑ ↑ 1 XEQ1 for step (a). Key T ↑ ΔG°_T B to display $-\Delta(G^\circ-H^\circ_{298})/RT$ with R in register 4 and ΔH°_{298} in register 5.

LBLB CHS RCL5 + RCL4 / x⇄y / STO IND6 RTN

Steps (c) and (d) are used unchanged. Before step (9) is used to calculate K, RCL4 ST/5 to convert ΔH°_{298} to $\Delta H^\circ_{298}/R$.

APPENDIX IIA (for Chapter II)

by Susie Hahn

1) Program CH

Program CH provides values of the Chebyshev polynomial terms $C_n(\bar{x})$ for n=1 to 4 and for $\bar{x}$=0 to N when N+1 data are to be inserted in program CB. C_0 = 1, $C_1 = 1 - 2\bar{x}/N$, $C_2 = 1 + 6\bar{x}(\bar{x}-N)/N(N-1)$ and additional terms can be calculated using the recurrence relation for a given $\bar{x}$:

$$C_{n+1} = [(2n+1)(N-2\bar{x})C_n - n(N+n+1)C_{n-1}]/(n+1)(N-n).$$

The sequence of calculations in program CH is outlined in the following flow chart. When the number of data points, N+1, is followed by XEQ 'CH', 1 is subtracted, N is stored in R00, N/2 is stored in R1, N/2 is compared with the integer value of N/2 to determine if N is even or odd. If odd, the calculation goes to LBL4, flag 1 is cleared, and the calculation proceeds to LBL5. If even, flag 1 is set and LBL5 is initiated. LBL5 calculates the last C_n register number, q = 20+4INT(N/2), which is stored in R4 and the C_n index number, 21 + q/1000, which is stored in R2. The form of the C_n index number is $21.\frac{q}{1000}$, where the integral part, 21, signifies the first register in which the first value of C_n will be stored and the fractional part, $\frac{q}{1000}$, signifies the last register in which the last value of C_n will be stored. Next, the C_n index number is reduced by one to accomodate the increment by 1 which occurs in LBL00, and this number is stored in R3. SIZE = q+2+N is determined and displayed to indicate the minimum number of registers required. The $\bar{x}$ index = 0 is stored in R7.

LBL00 increments the C_n index in R3 by 1 and the calculation jumps to LBL3 if the C_n index in R3 is not greater than q. At this point, C_n index = $21.\frac{q}{1000}$ and since 21 $\not>$ q, the calculation goes to LBL3.

LBL3 stores the n index of 1 in R6 and increments the $\bar{x}$ index in R7 by 1 and stores this new $\bar{x}$. Then $C_1(1) = 1 - 2/N$ is calculated and stored indirectly in R21 as directed by the C_n index in R3.

LBL1 then uses the recurrence relation to calculate $C_2(1) = [2C_1(1)(N/2 - \bar{x})(2n + 1) - n(N + n + 1)C_0(1)]/(n + 1)(N - n)$ with $n = \bar{x} = C_0(1) = 1$. The calculation proceeds as follows:

$C_1(1)$ → already in X stack position from LBL3
2
* $2C_1(1)(N/2 - \bar{x})(2n+1)$
$N/2 - \bar{x}$ → RCL1 - RCL7   in X to Y → X<>Y
*
n*2+1 → RCL6 x 2 + 1
*
6*(N+n+1) → RCL6 x (RCL00 + RCL6 + 1) $n(N+n+1)C_0(1)]$ in X
-   X - Y in X
N-n → RCL00 - RCL6 $\dfrac{[2C_1(1)(N/2 - \bar{x})(2n+1) - n(N+n+1)C_0(1)]}{N - n}$
/
n+1 → RCL6 + 1
/ $C_2(1) = \dfrac{[2C_1(1)(N/2 - \bar{x})(2n+1) - n(N+n+1)C_0(1)]}{(n+1)(N-n)}$

Then the C_n index in R3 is incremented by 1, which at this point is $22.\frac{q}{1000}$, so that $C_n(1)$ is stored indirectly in R22 as directed by the C_n index in R3.

The next sequence of steps brings 4, the maximum number for n, into the X stack position, recalls the n index in R6 and increments it by 1, which places the new value of n in the X stack position and pushes 4 into the Y stack position. Then, the new value of n is compared to the maximum value for n, 4. If $n \neq 4$ (and at this point n=2), the next step is skipped and the new value of n is stored in R6. The C_n index in R3 is reduced by 1 to obtain the previous value, which at this point becomes $21.\frac{q}{1000}$, so that the number C_1 corresponding to the register of this index can be retrieved.

The next sequence of steps arranges the stack positions as follows:

T	$C_2(1)$		4		n index		n index
Z	4	$\xrightarrow{R\uparrow}$	n index	$\xrightarrow{\text{RCL IND3}}$	1	$\xrightarrow{X<>Y}$	1
Y	n index		1		$C_2(1)$		$C_1(1)$
X	1		$C_2(1)$		$C_1(1)$		$C_2(1)$

Then, the C_n index in R3 is incremented by 1, which at this point becomes $22.\frac{q}{1000}$, to restore the index to the proper value in its sequence.

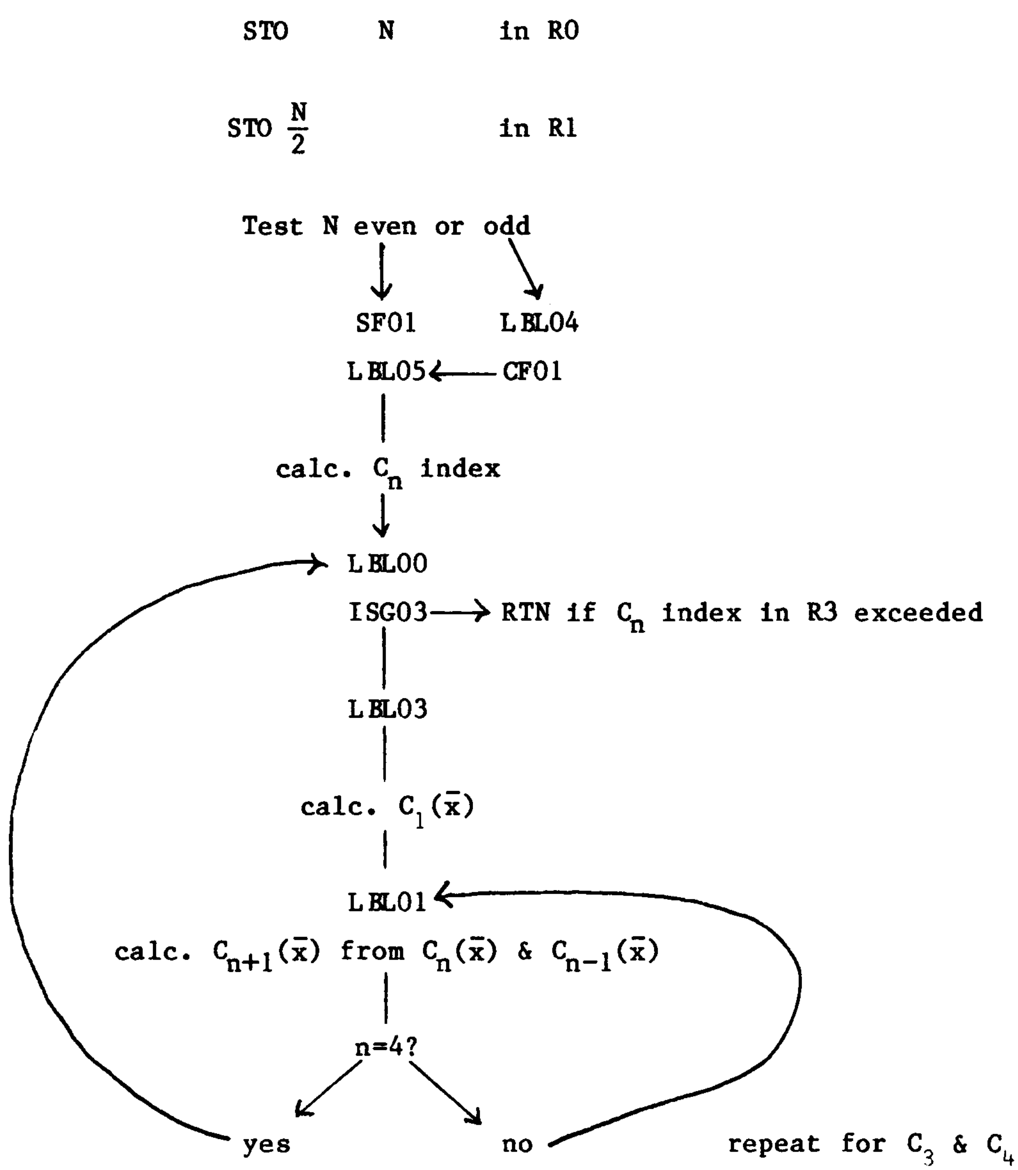
The stacks are arranged in the way shown above when LBL1 is again executed. LBL1 repeats the process in a similar manner but with n=2 in R6 and $C_2(1)$ in the X stack position to calculate $C_3(1)$ from the recurrence relation. When $C_3(1)$ is calculated, the C_n index is incremented by 1 so that $C_3(1)$ can be stored in the next available register, which at this point is R23. Since n $\neq$ 4 yet, the stacks are again arranged so that $C_3(1)$ is in the X stack position and $C_2(1)$ is in the Y stack position. Then GTO 01 again executes LBL1 to repeat the calculation with n=3 in R6 and $C_3(1)$ in X to determine $C_4(1)$ which is stored in the next available register, R24.

At this point, n = 4 so that the test condition x=y? is true. Therefore, the program executes LBL00 which increments the C_n index by 1 to position the next available register, which at this point becomes R25, and then jumps to LBL3. LBL3 restores the n index in R6 to 1 and increments the $\bar{x}$ index in R7 by 1. Then, $C_1(2) = 1 - 2\bar{x}/N$, where $\bar{x} = 2$ at this point is calculated and stored indirectly in R25. The program repeatedly executes LBL1 in a manner analogous to the one previously described to calculate $C_2(2)$, $C_3(2)$, and $C_4(2)$. When the n index in R6 is incremented to 4, the X=Y? test sends the calculation back to LBL00, LBL3, and then LBL1 to calculate $C_n(\bar{x})$ for $\bar{x}$ in R7, one larger than the previous calculation, starting with n=1 again. The loop is repeated for each value of $\bar{x}$ in R7 until the return to LBL00 increments R3 beyond the limit for the storage of the $C_n(\bar{x})$ values. If ISG 03 in LBL00 is true, that is, the integer before the decimal point in the last calculated C_n index number is greater than q, then the next step in skipped and the calculation stops with a display of the initial C_n index = 21 + q/1000.

Putting the calculator in the USER mode and then pressing B executes label B. LBL B recalls the original C_n index and stores it in R3. Then LBL 2 is executed. The display is fixed to 0 to display only the register number part of the C_n index. Then, the C_n value corresponding to this register number is recalled and displayed, fixing to 2 decimal places. The register number is incremented by 1 and as long as the register number is less than the q value, the entire process is repeated until all the register numbers and their corresponding C_n values are displayed.

Putting the calculator in the USER mode and then pressing C with the printer attached prints the register numbers and their corresponding C_n values since the instruction $21.\frac{q}{1000}$ PRREGX prints the registers from 21 to q and the corresponding values in these registers.

N+1 XEQ CH



User B displays Reg. number & C_n value

User C prints Reg. number & C_n value

2) Program CB

Introduction:

The Chebyshev polynomials are so useful in treating data, that a detailed discussion is presented. A summary of the nomenclature, equations, and calculation procedures is presented here.

The Chebyshev polynomial, $C_0(\bar{x})=1$, $C_1(\bar{x})=1-2\bar{x}/N$, and $C_{n+1}=[(2n+1)(N-2\bar{x})C_n - n(N+n+1)C_{n-1}]/(n+1)(N-n)$, is orthogonal for discrete integer values of $\bar{x}$ from 0 to N. If x_i is the initial value of x and I is the regular interval between x values, N+1 data points are assigned integral $\bar{x}$ values from 0 to N where $\bar{x} = (x-x_i)/I$ and the data are fit to a polynomial of the type:

$$f(x) = c_0C_0(x) + c_1C_1(x) + c_2C_2(x) + c_3C_3(x) + c_4C_4(x)$$

A least square fit is used to fit the data, but because of the orthogonality of $C_n(\bar{x})$, cross terms are zero in a matrix used to solve the set of linear equations obtained by setting the partial derivatives of the squares of the deviations equal to zero. Thus, the coefficients, c_n, of the polynomial are readily calculated without solution of the matrix by the relation:

$$c_n = (f,C_n)/(C_n,C_n)$$

where

$$(f,C_n) = \sum_{\bar{x}=0}^{N} f(\bar{x})C_n(\bar{x})$$

and

$$(C_n,C_n) = \sum_{\bar{x}=0}^{N} [C_n(\bar{x})]^2 = \frac{(N+n+1)!\ (N-n)!}{(2n+1)\ (N!)^2}$$

If f(x) is desired as a function of $\bar{x}$, an expansion of the $C_n(\bar{x})$ values in powers of x by

$$C_n(\bar{x}) = \sum_{m=0}^{n} (-1)^m \frac{(n+m)!\ \bar{x}!\ (N-m)!}{(n-m)!\ (m!)^2\ (\bar{x}-m)!\ N!}$$

yields:

$$f(\bar{x}) = \sum_{m=0}^{n} \alpha_m(\bar{x})^m = \alpha_0 + \bar{x}[\alpha_1+\bar{x}(\alpha_2+\bar{x}(\alpha_3+\bar{x}(\alpha_4)))]$$

The α_m terms are calculated from the relation

$$\alpha_m = \sum_{n=m}^{n'} B_{nm}(f, C_n)$$

where the values of B_{nm} are calculated from

$$B_{31} = m(m+1) = 2$$

$$B_{41} = -\frac{(m+2)}{2}(3+m) = -6$$

$$B_{32} = -\frac{m}{2}(1+m) = -3$$

$$B_{42} = m^2+3m+1 = 11$$

$$B_{43} = -\frac{m}{2}(1+m) = -6$$

so that

$$\alpha_0 = \sum_{n=0}^{n'} \frac{(f, C_n)}{(C_n, C_n)}$$, where n' denotes a maximum n value of $n'=3$ or 4.

$$\alpha_1 = \frac{(f, C_4)}{(C_4, C_4)}[C_{41} - C_{42} + B_{31}C_{43} + B_{41}C_{44}]$$

$$+ \frac{(f, C_3)}{(C_3, C_3)}[C_{31} - C_{32} + B_{31}C_{33}] + \frac{(f, C_2)}{(C_2, C_2)}[C_{21} - C_{22}]$$

$$+ \frac{(f, C_1)}{(C_1, C_1)}[C_{11}]$$

$$\alpha_2 = \frac{(f, C_4)}{(C_4, C_4)}[C_{42} + B_{32}C_{43} + B_{42}C_{44}] + \frac{(f, C_3)}{(C_3, C_3)}[C_{32} + B_{32}C_{33}]$$

$$+ \frac{(f, C_2)}{(C_2, C_2)}[C_{22}]$$

$$\alpha_3 = \frac{(f, C_4)}{(C_4, C_4)}[C_{43} + B_{43}C_{44}] + \frac{(f, C_3)}{(C_3, C_3)}[C_{33}]$$

$$\alpha_4 = \frac{(f, C_4)}{(C_4, C_4)}[C_{44}]$$

where values of C_{nm} are calculated from:

$$\frac{C_{n,m+1}}{C_{n,m}} = \frac{(n+m+1)(m-n)}{(m+1)^2 \ (N-m)} \quad \text{starting with } C_{n0} = 1.$$

If f(x) is desired as a function of x, the conversion

$$f(x) = \sum_m \alpha_m(\bar{x})^m = \sum_m a_m x^m$$

can be made using the relations:

$$a_0 = -\alpha_3 \left(\frac{x_i}{I}\right)^2 + \alpha_2 \left(\frac{x_i}{I}\right)^2 - \alpha_1\left(\frac{x_i}{I}\right) + \alpha_0$$

$$a_1 = 3\alpha_3 \frac{(x_i)^2}{I^3} - 2\alpha_2\left(\frac{x_i}{I^2}\right) + \frac{\alpha_1}{I}$$

$$a_2 = -3\alpha_3 \frac{x_i}{I^3} + \frac{\alpha_2}{I^2}$$

$$a_3 = \frac{\alpha_3}{I^3}$$

$$a_4 = \frac{\alpha_4}{I^4}$$

The contributions of α_4 to a_0 to a_4 must be added to each a_m value if the quartic term is present. The α_4 term contributes to each a_m term from m = n to m = 0. For a given m and n, the contribution to a_m is

$$\frac{\alpha_m}{I^m} (x_i)^{m-n} \sum_{j=0}^{j_{max}} \frac{j-m}{j+1}$$

where j is a positive integer increasing from 0 to j_{max} = m - n - 1. The a_0 contribution is α_4/I^4. The a_0 contribution multiplied by $(-4x_i)$ yields the a_1 contribution. The a_1 contribution multiplied by $(-1.5x_i)$ yields the a_2 contribution. The a_2 contribution multiplied by $(-x_i/1.5)$ yields the a_3 contribution. Multiplication by $-\frac{1}{4}(x_i)$ yields the a_4 contritution.

Program CB first calculates the contribution of the quartic term $\frac{(f, C_4)}{(C_4, C_4)}$. A decision is made whether to retain the quartic term, and the remainder of the calculation can be carried out for a cubic or quartic fit. Due to the orthogonality of $C_n(\bar{x})$, the coefficients of the earlier terms are <u>not</u> changed if the quartic term is dropped for the Chebyshev polynomial. Also, the symmetry of the function reduces the calculation by half. The program indicates the degree of fit by calculating $(\hat{f}(\bar{x})-f(\bar{x}))$, the standard deviation of the mean $\sigma = \left(\frac{\Sigma\left(\hat{f}(\bar{x})-f(\bar{x})\right)^2}{(N-1)}\right)^{1/2}$, and the mean derivation $\sum|\hat{f}(\bar{x})-f(\bar{x})|/(N+1)$ where for each data point, $\hat{f}$ denotes the value calculated from the Chelyshev polynomial and f denotes the corresponding value of the input data.

Explanation of Program CB Steps:

To begin Program CB, key the first data point, f(0), and then XEQ 'CB'. The first step of the program stores 6.1 in register R4. This number, actually 6.100, is the (f,C_n) index number. Values for (f,C_n) are stored in R6 to R10; the 100 being a "dummy" counter test value. The next step recalls the C_n index number, $21.\frac{q}{1000}$, calculated from Program CH from R2 and stores it in R3. Then, f(0) is brought down from the z into the x stack position before the program jumps to LBL11.

LBL11 recalls the C_n index number, takes its fractional part, and multiplies by 1000 to yield q. Adding 1 to q yields the register in which f(0) will be stored. This quantity, q + 1, is stored in R5. The next sequence of steps calculates q + 1 + N (in R00), which is the register number for the last input data, f(N). Dividing this quantity by 1000 and adding this to the number previously stored in R5 yields the f index number, $q+1.\frac{q+1+N}{1000}$, where the input data $f(\bar{x})$ will be stored from R(q+1) to R(q+1+N). The stack is rolled down to restore f(0) to the x position before the program continues to LBL10.

LBL10 indirectly stores f(0) in R(q+1), directed by the f index number in R5. Then the f index is incremented by 1 to position the register for the next input data before the program execution returns to LBL CB right after the XEQ11 instruction. Since f(0) is still in the x stack position, it is stored in R6,7,8,9 and 10. The f(0) value can be stored directly in these (f,C_n) registers because f(0) $C_n(0)$ = f(0), since $C_n(0) = 1$. The return instruction stops the program and displays f(0).

Now, f(1) should be entered and then keying R/S resumes the program which executes LBL10. LBL10 indirectly stores f(1) in the next available f register and again increments the f index number to position the register for the next input data. The program returns to LBL 'CB' after the XEQ10 instruction and proceeds to LBL1.

LBL1, in conjunction with LBL2, calculates the (f,C_n) values for the first half of N+1 data points if N is odd or for the first half plus 1 of the data points if N is even, excluding, of course, the first data point, f(0). The first steps in LBL1 fill the stacks with the previously entered f(1) value since LBL2 uses it 4 times in the XEQ2 command. The next step indirectly adds f(1) to f(0) in R6, directed by the (f,C_n) index in R4 which at this point is 6.1. R6 will contain $\sum f(\bar{x})$ since $(f,C_0) = \sum f(\bar{x})C_0(\bar{x})$ and $C_0(\bar{x}) = 1$. Then, the (f,C_n) index is incremented by 1 to 7.1 to position the next (f,C_n) register for the value $f(1)C_1(1)$. Next, the program jumps to LBL2.

In LBL2, the stack is rolled down to remove the previous (f,C_n) value (there is no (f,C_n) value before the first XEQ2 in LBL1 but just $f(\bar{x})$) and bring $f(\bar{x})$ into the x stack position. $C_1(1)$ is indirectly recalled from R21 to the x stack position, directed by the C_n index in R3, which pushes f(1) into the y stack position. Then f(1) is multiplied by $C_1(1)$ and this value is indirectly added to the contents of R7, $f(0)C_1(0)$, and this sum is indirectly stored in R7, directed by the (f,C_n) index in R4 which is 7.1 at this point. The (f,C_n) index is incremented by 1 to 8.1, and the C_n index is also incremented by 1 to prepare to retrieve the next C_n value. Then program execution returns to LBL1 to XEQ2 a second time.

This time LBL2 removes the previous (f,C_n) value and returns f(1) to the x stack position. $C_2(1)$ is indirectly recalled from R22, directed by the C_n index in R3. Then, $f(1)xC_2(1)$ is calculated and indirectly added to the contents of R8, $f(0)C_2(0)$, and this sum is indirectly stored in R8, directed by the (f,C_n) index in R4 which is 8.1 at this point. The (f,C_n) index and the C_n index are both incremented by 1 before execution returns to LBL1 to XEQ2 a third time.

LBL2 removes the previous (f,C_n) value and brings f(1) back to the x stack position. $C_3(1)$ is indirectly recalled from R23, directed by the C_n index in R3. Then $f(1)xC_3(1)$ is calculated and indirectly added to the contents of R9, $f(0)C_3(0)$, and this sum is indirectly stored in R9, directed by the (f,C_n) index in R4 which is 9.1 at this point. The (f,C_n) index and the C_n index are both incremented by 1 before execution returns to LBL1 to XEQ2 a fourth time.

LBL2 again removes the previous (f,C_n) value and returns f(1) to the x stack position. $C_4(1)$ is indirectly recalled from R24, directed by the C_n index in R3. Then $f(1)xC_4(1)$ is calculated and indirectly added to the contents in R10, $f(0)C_4(0)$, directed by the (f,C_n) index in R4 which is 10.1 at this point. The (f,C_n) index is incremented to 11.1 and the C_n index is incremented to 25.q before execution returns to LBL1 after the fourth XEQ2.

By subtracting 5 from R4 in LBL1, the (f,C_n) index is restored to 6.1 so that the next (f,C_n) value, $f(2)C_0(2)$ may be added to the proper (f,C_n) register, R6. Rolling up the stack brings f(1) into the x stack position. RTN stops the program, displays f(1), and allows f(2) to be entered; program resumes by keying R/S to execute LBL10.

LBL10 indirectly stores f(2) in the next available f register, directed by the f index in R5 and this index is then incremented by 1 before execution returns to LBL1 right before the GTO1 command which brings the program to the beginning of LBL1.

In a manner completely analogous to that just described, LBL1 and LBL2 again calculate (f,C_n) values, adds these values to the proper (f,C_n) registers, and store these sums in their respective registers. Thus for f(2), the registers contain:

$f(2)\ \overline{+\ f(1)\ +\ f(0)}^{\text{previously in R6}}$	STO+IND04 or R6 with 6.1 in R4
$f(2)C_1(2) + \overline{f(1)C_1(1) + f(0)C_1(0)}^{\text{previously in R7}}$	STO+IND04 or R7 with 7.1 in R4
$f(2)C_2(2) + \overline{f(1)C_2(1) + f(0)C_2(0)}^{\text{previously in R8}}$	STO+IND04 or R8 with 8.1 in R4
$f(2)C_3(2) + \overline{f(1)C_3(1) + f(0)C_3(0)}^{\text{previously in R9}}$	STO+IND04 or R9 with 9.1 in R4
$f(2)C_4(2) + \overline{f(1)C_4(1) + f(0)C_4(0)}^{\text{previously in R10}}$	STO+IND04 or R10 with 10.1 in R4

At this point, immediately after the STO+IND04 instruction from the fourth execution of LBL2, the (f,C_n) index and the C_n index are again incremented by 1. Assume now that half plus one (if N=4) or that half (if N=5) of the N + 1 data points, that is, $f(\text{INT } \frac{N}{2})$ have been keyed in thus far. Then the C_n index will have been exceeded by the last incrementation to $(q+1).\frac{q}{1000}$. Thus, the next step, RTN, is skipped and the stack is rolled down to remove the previous (f,C_n) value and return f(2) to the x stack position. The C_n index, $(q+1).\frac{q}{1000}$, is recalled from R3 and the integral part is taken and stored back into R3. Then the stack is rolled down to return f(2) to the x stack position before the C_n index in R3 is decremented by 1 to yield q in R3, the position of the last C_n value. Due to the symmetry of the Chebyshev polynomial function, the program can run backwards through the C_n values to obtain the remaining (f,C_n) values, and the DSE 03 instruction recalls the C_n values from the last to the first C_n register.

The next step, FS?01, tests whether N is even (flag 1 was set in Program CH if N was even). If so, the program jumps to LBL4. If N is even, in keeping with the symmetry, the last 4 C_n values are not needed to calculate the remaining (f,C_n) values:

$$\begin{array}{rl} \text{keyed in} & \left\{\begin{array}{l} f(0) \times C_n(0) \\ f(1) \times C_n(1) \\ f(2) \times C_n(2) \end{array}\right. \\ \text{thus} & \\ \text{far:} & \end{array} \quad \text{-- last 4 } C_n \text{ values are used only once}$$

$$\begin{array}{l} f(3) \times C_n(1) \\ f(4) \times C_n(0) \end{array}$$

Therefore, LBL4 decreases the C_n index in R3 by 4, positioning the proper register, R(q-4), for the next (f,C_n) calculation. Then the stack is rolled down to display f(2) before the program returns to LBL2 and stops to display f(2). Then f(3) is entered and R/S keyed to resume the program execution with LBL10.

LBL10 indirectly stores f(3) in the proper f register, guided by the f index number in R5. Then, this index is incremented by 1 before the program returns to LBL2 to go to LBL5.

LBL5 fills the x, y, and z stack positions with f(3) and then jumps to LBL9. LBL9 indirectly recalls the proper C_n value, $C_4(1)$, in the case with N=4, directed by the C_n index in R3. Then this index is decremented by one to position to the correct C_n value for the next time LBL9 is executed. Then with f(3) in the y stack position and $C_4(1)$ in the x stack position, the two numbers are multiplied, yielding $f(3)C_4(1)$. The program returns to LBL5, adds the last (f,C_n) value to the sum of (f,C_n) values previously in R10, and stores this new sum, $f(3)C_4(1) + f(2)C_4(2) + f(1)C_4(1) + f(0)C_4(0)$, in R10.

Then the stack is rolled down to bring f(3) into the x stack position before the program jumps to LBL6.

LBL6 rearranges the stack as follows:

T	$f(3)C_4(1)$	f(3)	f(3)	f(3)	f(3)	−f(3)
Z	f(3)	f(3)	f(3)	f(3)	−f(3)	f(3)
		STO T →	STO Y →	CHS →	STO Z →	RDN →
Y	f(3)	f(3)	f(3)	f(3)	f(3)	−f(3)
X	f(3)	f(3)	f(3)	−f(3)	−f(3)	f(3)

Then f(3) is added to the contents of R6 and stored, to yield f(3) + f(2) + f(1) + f(0) in R6. Then, the program returns to LBL5, after the XEQ6 command, with the stacks arranged as shown after the RDN instruction in LBL6.

The importance of the alternating signs comes about from the symmetry of the Chebyshev polynomial function. If a symmetry plane is drawn half way between the $\bar{x}$'s for the $C_n(\bar{x})$ values, the values above the plane are equal to their corresponding "mirror images" below the plane, except that for n=1 and 3, the signs of the symmetrical C_n values are opposite. For example, if N=4 the C_n values are:

$\bar{x}$ \ **n**	0	1	2	3	4	
0	1	1	1	1	1	
1	1	$^1/2$	$-^1/2$	−2	−4	
2	1	0	−1	0	6	- - - - symmetry plane
3	1	$-^1/2$	$-^1/2$	2	−4	
4	1	−1	1	−1	1	

"mirror images" have opposite signs
but same absolute value

Instead of changing the signs of the proper C_n values, the signs of the corresponding $f(\bar{x})$ values will be changed to calculate (f,C_n) values.

Back in LBL5, the stacks are rolled down to yield:

T	f(3)
Z	-f(3)
Y	f(3)
X	-f(3)

Then LBL9 is executed which indirectly recalls the proper C_n value, $C_3(1)$ in the case with N=4, guided by the C_n index in R3. Then this index is again decremented by 1. Now, because N=3, -f(3) is in the Y stack position and $C_3(1)$ is in the X stack position so multiplication yields $-f(3)C_3(1)$. The program returns to LBL5, adds the last (f,C_n) value to the sum of (f,C_n) values previously in R9, and stores this sum, $-f(3)C_3(1) + f(2)C_3(2) + f(1)C_3(1) + f(0)C_3(0)$ in R9. Then the stack is rolled down to bring f(3) into the X stack position before the program jumps again to LBL9. LBL9 indirectly recalls the proper C_n value, $C_2(1)$ in the case with N=4, directed by the C_n index in R3. This index is then decremented by 1. With f(3) now in the Y stack position and $C_2(1)$ in the X stack position, $f(3)C_2(1)$ is calculated and the program returns to LBL5 to add this value to R8, and stores the sum, $f(3)C_2(1) + f(2)C_2(2) + f(1)C_2(1) + f(0)C_2(0)$, in R8. Then the stack is rolled down to bring -f(3) into the X stack position before execution jumps again to LBL9. LBL9 recalls the proper C_n value, $C_1(1)$ in the case with N=4, directed by the C_n index in R3. This index is decremented by 1. With -f(3) now in the Y stack position, since n=1, and $C_1(1)$ in the X stack position $-f(3)C_1(1)$ is calculated and the program returns to LBL5 to add this value to the contents of R7, and stores this sum, $-f(3)C_1(1) + f(2)C_1(2) + f(1)C_1(1) + f(0)C_1(0)$, in R7. Next, the stack is rolled up to display -f(3) when the program halts. At this point, f(4) is keyed in and then R/S so that the program returns with LBL10.

LBL10 indirectly stores f(4) in the proper f register, guided by the f index in R5. Then the f index is incremented. If f(4) were not the last data point, the program loops again through LBL5, LBL9, LBL6, and LBL10 in an analogous manner to that just described to determine and store the $\sum_{\bar{x}=0}^{N-1} f(\bar{x})C_n(\bar{x})$ values, up to N-1. However, if f(4) is the last data point, as it has been formerly assumed, then this last incrementation exceeded the f index. Therefore, the RTN instruction is skipped and the program executes LBL6.

LBL6 again rearranges the stack as follows:

T	-f(3)		f(4)		f(4)		f(4)		f(4)		-f(4)
Z	$f(3)C_1(1)$		$f(3)C_1(1)$		$f(3)C_1(1)$		$f(3)C_1(1)$		-f(4)		f(4)
		STO T →		STO Y →		CHS →		STO Z →		RDN →	
Y	-f(3)		-f(3)		f(4)		f(4)		f(4)		-f(4)
X	f(4)		f(4)		f(4)		-f(4)		-f(4)		f(4)

Then, f(4) is added to the contents of R6, and the sum f(4) + f(3) + f(2) + f(1) + f(0) is stored in R6. The program returns to LBL10. Since $C_n(N) = \pm 1$, $f(N)C_n(N) = \pm f(N)$ so that f(N) can be directly stored in R8 and R10 and -f(N) can be directly stored in R7 and R9, for which n=1 and 3, respectively. Thus, f(4) (still in the X stack position) is added to the contents of R10, and the sum, $f(4)C_4(4) + f(3)C_4(1) + f(2)C_4(2) + f(1)C_4(1) + f(0)C_4(0)$ is stored in R10. Then the stack is rolled down to bring -f(4) into the X position and this is added to the contents of R9, and the sum, $-f(4)C_3(4) - f(3)C_3(1) + f(2)C_3(2) + f(1)C_3(1) + f(0)C_3(0)$, is stored in R9. The stack is rolled down again to bring f(4) into the X position and this is added to the contents of R8, and the sum, $f(4)C_2(4) + f(3)C_2(1) + f(2)C_2(2) + f(1)C_2(1) + f(0)C_2(0)$, is stored in R8. Again, the stack is rolled down to bring -f(4) into the X position and this is added to the contents of R7, and the sum, $-f(4)C_1(4) -f(3)C_1(1) + f(2)C_1(2) + f(1)C_1(1) + f(0)C_1(0)$ is stored in R7.

If, in LBL2, N was odd (for instance N=5), then the FS?01 test would be false and the next step, XEQ4, would be skipped. If N is odd, in keeping with the symmetry of the Chebyshev function, the last four C_n values are used to calculate the remaining (f,C_n) values:

$$\text{keyed in thus far: } \left\{\begin{array}{l} f(0) \times C_n(0) \\ f(1) \times C_n(1) \\ f(2) \times C_n(2) \end{array}\right. \qquad \begin{array}{l} f(3) \times C_n(2) \\ f(4) \times C_n(1) \\ f(5) \times C_n(0) \end{array}$$

last 4 C_n values are used for the upcoming (f,C_n) value

This execution proceedes immediately with LBL10 after f(3). R/S is keyed in, skipping LBL4 which decrements the C_n index by 4. Then the program loops through LBL5, LBL9, LBL6, and LBL10, in an analogous manner to that previously described to determine and store the $\sum_{x=0}^{N} f(\bar{x})C_n(\bar{x})$ values in their proper registers.

Next, whether N was even or odd, LBL17 is executed. LBL17 calculates the part of the α_m terms, involving the quartic term. First, 0, the m index, is stored in R1 and 4, the n index, is stored in R4. Then, LBL13 is executed.

LBL13 calculated the reciprocal values of (C_n,C_n). At this point, since n = 4, it calculates $1/(C_4,C_4)$ as follows:

$(N!)^2$ -- R00, FACT, X^2

$(N - n)!$ -- R00 - R4, FACT

$(N!)^2/(N - n)!$ -- /

$(N + n + 1)!$ -- RCL00 + RCL4 + 1, FACT

$(N!)^2/(N + n + 1)!\ (N - n)!$ -- /

$(2n + 1)$ -- RCL4 * 2 + 1

$(2n + 1)(N!)^2/(N + n + 1)!\ (N - n)!$ -- *

Then execution returns to LBL17, with $1/(C_4, C_4)$ in the X stack position, where (f, C_4) is recalled from R10 and multiplied to $1/(C_4, C_4)$ to yield $\frac{(f, C_4)}{(C_4, C_4)}$ which is stored in R11. The program stops to display this ratio which is the contribution of the quartic term to $\hat{f}(0)$ and $\hat{f}(N)$ for which $C_4 = 1$. As $C_4(\bar{x})$ is usually less than 1, the display indicates the maximum error if the quartic term is dropped. If the quartic term is to be retained, key R/S; if it is to be dropped, press USER A.

If R/S is pressed, flag 0, which indicates the quartic term is dropped, is cleared. One is placed into the X stack position before LBL14 is executed because LBL14 calculates $\frac{(C_{n,m+1})}{(C_{n,m})} \times (C_{n,m})$ where $C_{n,m}$ is obtained from the previous run through of LBL14. However, for the first execution of LBL14 in the sequence, $C_{n,m} = 1$, since $n = 4$ and $m = 0$ and $C_{4,0} = 1$.

LBL14 calculates $\frac{(C_{n,m+1})}{(C_{n,m})}$ in the following manner:

$(m + 1)^2$ -- (RCL1 + 1), x^2

$(m + 1 + n)$ -- LASTx + RCL4

$(m + 1)^2/(m + 1 + n)$ -- /

$(n + m + 1)/(m + 1)^2$ -- 1/X

$(m - n)$ -- RCL1 - RCL4

$(n + m + 1)(m - n)/(m + 1)^2$ -- *

$(N - m)$ -- RCL00 - RCL1

$\frac{(C_{n,m+1})}{(C_{n,m})} = (n + m + 1)(m - n)/(m + 1)^2(N - m)$ -- /

At this point, $n = 4$ and $m = 0$ so that $\frac{(C_{4,1})}{(C_{4,0})}$ was calculated. Then, with $C_{4,0} = 1$ now in the Y stack position, $\frac{(C_{4,1})}{(C_{4,0})}(C_{4,0})$ is calculated so that execution returns to LBL17 with $C_{4,1}$ in the X stack position.

In LBL17, $C_{4,1}$ is stored in R20. The m index in R1 is incremented by 1 so that at this point, $m = 1$. Then the stack is rolled down to restore $C_{4,1}$ to the X stack position before LBL14 is executed again.

LBL14 again calculates $\frac{(C_{n,m+1})}{(C_{n,m})}$, this time with n = 4 and m = 1. Since $C_{4,1}$ is in the Y stack position, $\frac{(C_{4,2})}{(C_{4,1})}(C_{4,1}) = C_{4,2}$ is calculated. Then the program returns to LBL17 with $C_{4,2}$ in the X stack position.

In LBL17, $C_{4,2}$ is stored in R19. Then $C_{4,2}$ is subtracted from the contents of R20, so that $C_{41}-C_{42}$ is stored in R20. Again, the m index in R1 is incremented by 1 to m=2. Then the stack is rolled down to restore $C_{4,2}$ to the X stack position before LBL14 is executed.

This time, LBL14 calculates $\frac{(C_{4,3})}{(C_{4,1})}$ and then $\frac{(C_{4,3})}{(C_{4,1})}(C_{4,1}) = C_{4,3}$. The program returns to LBL17 with $C_{4,3}$ in the X stack position.

In LBL17, $C_{4,3}$ is stored in R18. Multiplying $C_{4,3}$ by 2 yields $B_{31}C_{43}$ since $B_{31} = 2$. Adding $B_{31}C_{43}$ to R20 yields $C_{41} - C_{42} + B_{31}C_{43}$ in R20. Again, the m index in R1 is incremented by 1 to m = 3. Then $C_{4,3}$ is restored to the X stack position before LBL14 is executed by recalling R18.

At this point, LBL14 calculates $\frac{(C_{4,4})}{(C_{4,3})}$ and then $\frac{(C_{4,2})}{(C_{4,3})}(C_{4,3}) = C_{4,2}$. The program returns to LBL17 with $C_{4,4}$ in the X stack position.

In LBL17, the contents of R20 are recalled and then the X and Y stack positions are interchanged:

$$\begin{array}{lll} Y \quad C_{4,4} & & C_{41} - C_{42} + B_{31}C_{43} \\ & \xrightarrow{\underline{X \lessgtr Y}} & \\ X \quad C_{41} - C_{42} + B_{31}C_{43} & & C_{4,4} \end{array}$$

Then $C_{4,4}$ is stored in R20. $C_{4,4}$ is multiplied by 6 and $6C_{4,4}$ is subtracted from $C_{41} - C_{42} + B_{31}C_{43}$ to yield $C_{41} - C_{42} + B_{31}C_{43} + B_{41}C_{44}$ since $B_{41} = -6$. Then this value is multiplied to the contents of R11, $(f,C_4)/(C_4,C_4)$, to yield $\frac{(f,C_4)}{(C_4,C_4)}[C_{41} - C_{42} + B_{31}C_{43} + B_{41}C_{44}]$ which is stored in R12. The next series of steps brings C_{42} in R19 to the Z stack position, C_{43} in R18 to the Y stack position, and 3 to the X stack position. Multiplying yields $3C_{43}$ and then subtracting yields $C_{42} - 3C_{43}$ or $C_{42} + B_{32}C_{43}$ since $B_{32} = -3$. The last value is pushed into the Z stack position as $C_{4,4}$ in R20 is placed in the Y stack position and 11 is placed in the X stack position. Multiplying yields $11C_{44}$ and adding yields $C_{42} + B_{32}C_{43} + B_{42}C_{44}$ since $B_{42} = 11$. Then this value is multiplied to the contents of R11, to yield $\frac{(f,C_4)}{(C_4,C_4)}$ $[C_{42} + B_{32}C_{43} + B_{42}C_{44}]$ which is stored in R13. The next series of steps brings C_{43} in R18 to the Z stack position, C_{44} in R20 to the Y stack position, and 6 to the X stack position. Multiplying yields $6C_{44}$ and then subtracting yields $C_{43} - 6C_{44}$ or $C_{43} + B_{43}C_{44}$ since $B_{43} = -6$. Then this value is multiplied to the contents of R11, to yield $\frac{(f,C_4)}{(C_4,C_4)}[C_{43} + B_{43}C_{44}]$ which is stored in R14. The contents of R20, C_{44}, are multiplied to the contents of R11, to yield $\frac{(f,C_4)}{(C_4,C_4)}[C_{44}]$ which is stored in R15. Now the quartic term of α_0 is stored in R11, that of α_1 is in R12, that of α_2 is in R13, that of α_3 is in R14, and that of α_4, which consists only of the quartic term, is in R15, that is, R15 contains α_4.

If, after the program halts in LBL17 to display the quartic term error, the decision is made to drop the quartic term, key USER A. LBL A stores the display, $\frac{(f,C_4)}{(C_4,C_4)}$, in R20. Then it clears R11, 12, 13, and 14 since the STO+ command will be used with these registers to determine α_0, α_1, α_2, and α_3. Also, if R/S was first keyed and then the decision is made to drop the quartic term, this step clears the quartic term contributions to the α_m values, which were previously calculated and stored in R 11 through R15 by the steps in LBL17 following R/S. (R15 is not cleared because it contains α_4 which is no longer required so that this register is never again recalled to use for the cubic fit.) Then, flag 0 is set to indicate that the quartic term has been dropped.

Next, LBL7 is executed whether the quartic term has been retained or not. LBL7 calculates α_0, α_1, α_2, α_3, and α_4 if the quartic term is retained. First, m = 0 is stored in R1 and n = 3 is stored in R4. One is placed in the X stack position since $C_{3,0} = 1$. Then LBL14 is executed.

Once again, LBL14 calculates $\frac{(C_{m,n+1})}{(C_{m,n})}$, or at this point $\frac{(C_{3,1})}{(C_{3,0})}$. Then $\frac{(C_{3,1})}{(C_{3,0})}(C_{3,0}) = C_{3,1}$ is calculated. The program returns to LBL7 with $C_{3,1}$ in the X stack position.

In LBL7, $C_{3,1}$ is stored in R19. Then the m index is increased by one to m = 1. The stack is rolled down to bring $C_{3,1}$ back in the X stack position before LBL14 is executed again.

This time, LBL14 calculates $\frac{(C_{3,2})}{(C_{3,1})}$ and then $\frac{(C_{3,2})}{(C_{3,1})}(C_{3,1}) = C_{3,2}$. The program returns to LBL7 with $C_{3,2}$ in the X stack position.

In LBL7, $C_{3,2}$ is subtracted from the contents of R19 so that $C_{31} - C_{32}$ is stored in R19. Then $C_{3,2}$ is stored in R17. The m index is incremented to m = 2. Then $C_{3,2}$ is returned to the X stack position before execution proceeds to LBL14.

LBL14 calculates $\frac{(C_{3,3})}{(C_{3,2})}$ and then $\frac{(C_{3,3})}{(C_{3,2})}(C_{3,2}) = C_{3,3}$. The program returns to LBL7 with $C_{3,3}$ in the X stack position.

In LBL7, $C_{3,3}$ is stored in R18. Then $C_{3,7}$ is multiplied by 2 and added to the contents of R19 to yield the sum $C_{31} - C_{32} + B_{31}C_{33}$, since $B_{31} = 2$. Then execution jumps to LBL13.

As before, LBL13 calculates $1/(C_n, C_n)$, but this time with n = 3 so that $1/(C_3, C_3)$ is calculated. The program then returns to LBL7.

In LBL7, $1/(C_3,C_3)$ is multiplied to (f,C_3) in R9 to yield $\frac{(f,C_3)}{(C_3,C_3)}$ which is stored in R19. This is added to the contents of R11 so that the sum $\frac{(f,C_4)}{(C_4,C_4)} + \frac{(f,C_3)}{(C_3,C_3)}$ is stored in R11. However, if the quartic term was dropped the first term would be zero. Since $C_{31}-C_{32}+B_{31}C_{33}$ is in the Y stack position and $\frac{(f,C_3)}{(C_3,C_3)}$ is still in the X position, multiplying gives $\frac{(f,C_3)}{(C_3,C_3)}[C_{31} - C_{32} + B_{31}C_{33}]$. This term is added to the contents of R12 to yield $\frac{(f,C_4)}{(C_4,C_4)} + [C_{41} - C_{42} + B_{41}C_{44}] + \frac{(f,C_3)}{(C_3,C_3)}[C_{31} - C_{32} + B_{31}C_{33}]$ which is stored in R12. Again, the first term would be zero for a cubic fit. The next series of steps places

C_{32} in the Z stack position, C_{32} in the Y position and 3 in the X position. Multiplying yields $3C_{33}$ and then subtracting yields $C_{32} - 3C_{33}$ or $C_{32} + B_{32}C_{33}$ since $B_{32} = -3$. This is multiplied by $\frac{(f,C_3)}{(C_3,C_3)}$ in R19 to yield $\frac{(f,C_3)}{(C_3,C_3)}$ $[C_{32} + B_{32}C_{33}]$ which is added to the contents of R13 to yield the sum $\frac{(f,C_4)}{(C_4,C_4)}[C_{42} + B_{32}C_{43} + B_{42}C_{44}] + \frac{(f,C_3)}{(C_3,C_3)}[C_{32} + B_{32}C_{33}]$ which is stored in R13. For a cubic fit, the first term would be zero. Then C_{33} in R18 is multiplied to $\frac{(f,C_3)}{(C_3,C_3)}$ in R19. This is then added to the contents of R14 to yield the sum $\frac{(f,C_4)}{(C_4,C_4)}[C_{43} + B_{43}C_{44}] + \frac{(f,C_3)}{(C_3,C_3)}C_{33}$ which is stored in R14. This sum is α_3. Again, the first term would be zero if the quartic term had been dropped. Next, the n index in R4 is decremented by 1 to n = 2 and the m index in R1 is decremented to 0. One is placed in the X stack position since $C_{2,0}$ = 1 before the program jumps to LBL14.

LBL14 calculates $\frac{(C_{2,1})}{(C_{2,1})}$ and then $\frac{(C_{2,1})}{(C_{2,0})}(C_{2,0}) = C_{2,1}$. The program returns to LBL7 with $C_{2,1}$ in the X stack position.

In LBL7, $C_{2,1}$ is stored in R19. The m index is incremented to m=1 and then $C_{2,1}$ is restored to the X stack position before LBL14 is executed again.

LBL14 calculates $\frac{(C_{2,2})}{(C_{2,1})}$ and then $\frac{(C_{2,2})}{(C_{2,1})}(C_{2,1}) = C_{2,2}$. The program returns to LBL7 with $C_{2,2}$ in the X stack position.

In LBL7, $C_{2,2}$ is stored in R17. Then $C_{2,2}$ is subtracted from C_{21} in R19 and $C_{21} - C_{22}$ is stored in R19 before execution jumps to LBL13.

LBL13 again calculates $1/(C_n,C_n)$ with n = 2. The program then returns to LBL7.

In LBL7, $1/(C_n,C_n)$ is multiplied to (f,C_2) in R8. Then $\frac{(f,C_2)}{(C_2,C_2)}$ is added to the contents of R11 to yield the sum $\frac{(f,C_4)}{(C_4,C_4)} + \frac{(f,C_3)}{(C_3,C_3)} + \frac{(f,C_2)}{(C_2,C_2)}$. For a cubic fit, the first term would be zero. $\frac{(f,C_2)}{(C_2,C_2)}$ is stored in R18 and then multiplied by $(C_{21} - C_{22})$ in R19. Next, this product is added to the contents of R12 to yield the sum, $\frac{(f,C_4)}{(C_4,C_4)}[C_{41} - C_{42} + B_{31}C_{43} + B_{41}C_{44}] + \frac{(f,C_3)}{(C_3,C_3)}[C_{31} -$

$C_{32} + B_{31}C_{33}] + \frac{(f,C_2)}{(C_2,C_2)}[C_{21} - C_{22}]$, which is stored in R12. Again, the first term would be zero for a cubic fit. Now, $C_{2,2}$ in R17 is multiplied to $\frac{(f,C_2)}{(C_2,C_2)}$ in R18. This product is added to the contents of R13 to yield the sum, $\frac{(f,C_4)}{(C_4,C_4)}$ $[C_{42} + B_{32}C_{43} + B_{42}C_{44}] + \frac{(f,C_3)}{(C_3,C_3)}[C_{32} + B_{32}C_{33}] + \frac{(f,C_2)}{(C_2,C_2)}C_{22}$ which is stored in R13. This sum is α_2. Once again, the first term would be zero if the quartic term had been dropped. Finally, the n index in R4 is decremented by 1 to n=1, the m index in R1 is decremented to 0, and 1 is placed in the X stack position since $C_{1,0} = 0$, before the program jumps to LBL14.

LBL14 calculates $\frac{(C_{1,1})}{(C_{1,0})}$ and then $\frac{(C_{1,1})}{(C_{1,0})}(C_{1,0}) = C_{1,1}$. Execution returns with $C_{1,1}$ in the X stack position to LBL7 which immediately executes LBL13.

LBL13 calculates $1/(C_n,C_n)$ for n=1. The program returns to LBL7.

In LBL7, $1/(C_{1,1})$ is multiplied to (f,C_1) in R7 and the product is added to the contents of R11 to yield $\frac{(f,C_4)}{(C_4,C_4)} + \frac{(f,C_3)}{(C_3,C_3)} + \frac{(f,C_2)}{(C_2,C_2)} + \frac{(f,C_1)}{(C_1,C_1)}$ which is stored in R11. Again, the first term would be zero if the quartic term had been dropped. Then multiplication with $C_{1,1}$ in the Y stack position and $\frac{(f,C_1)}{(C_1,C_1)}$ in the X position yields $\frac{(f,C_1)}{(C_1,C_1)}(C_{1,1})$ which is added to the contents of R12 to yield the sum, $\frac{(f,C_4)}{(C_4,C_4)}[C_{41} - C_{42} + B_{31}C_{43} + B_{41}C_{44}] +$ $\frac{(f,C_3)}{(C_3,C_3)}[C_{31} - C_{32} + B_{31}C_{33}] + \frac{(f,C_2)}{(C_2,C_2)}[C_{21} - C_{22}] + \frac{(f,C_1)}{(C_1,C_1)}[C_{1,1}]$. This sum is α_1. Again, the first term would be zero for a cubic fit. Next, (f,C_0) in R6 is placed in the Z stack position, N in R00 is placed in the Y position, and 1 is placed in the X position. Addition yields N+1, and then division yields $(f,C_0)/(N+1)$. (N+1) is the value of $1/(C_n,C_n)$ for n=0. Then, $(f,C_0)/(C_0,C_0)$ is added to the contents of R11 to yield $\frac{(f,C_4)}{(C_4,C_4)} + \frac{(f,C_3)}{(C_3,C_3} + \frac{(f,C_2)}{(C_2,C_2)} + \frac{(f,C_1)}{(C_1,C_1)}$ $+ \frac{(f,C_0)}{(C_0,C_0)}$. This sum in R11, minus the $\frac{(f,C_4)}{(C_4,C_4)}$ contribution if the quartic term had been dropped, is α_0. At this point, the program stops to display α_0. Then keying SST recalls α_1 in R12 to the display, keying SST once more recalls α_2 in R13 to the display, SST again recalls α_3 in R14 to the display, and if the quartic term was retained, keying SST a final time recalls α_4 in R15 to the display. If the quartic term had been dropped, the α_m value displayed would not contain the quartic term contribution and, of course, α_4 would not exist.

LBL D tabulates the closeness of fit of the Chebyshev polynomial equation to the data for each data point. To execute LBL D, turn off the calculator, attach the printer in manual mode, and key USER D. LBL D fixes the number of places after the decimal to 3. Then it recalls N in R00 and adds 1. Since at this point, the f index in R5 is exceeded by 1, subtracting 5 from this index positions the f registers to f(0). Then α_0 is recalled from R11 before execution jumps to LBL16.

LBL16 holds instructions for the printer. First, f(0) is recalled indirectly by the f index in R5. A copy of f(0) in the X register is accumulated into the print buffer when the instruction, ACX, is given. 3 SHPCHR tells the printer to skip 3 spaces on a line. The next two steps rearrange the stacks as follows:

$$\begin{array}{llcccl} T & N+1 & & 3 & & N+1 \\ Z & \alpha_0 & \xrightarrow{X<>T} & \alpha_0 & \xrightarrow{RDN} & 3 \\ Y & f(0) & & f(0) & & \alpha_0 \\ X & 3 & & N+1 & & f(0) \end{array}$$

Subtracting yields $\alpha_0 - f(0)$ which equals $\hat{f}(0) - f(0)$, where $\hat{f}$ denotes the calculated value and f denotes the data input value, since $\hat{f}(0) = \alpha_0$, using the general equation, $\hat{f}(\bar{x}) = \alpha_0 + \bar{x}[\alpha_1 + \bar{x}(\alpha_2 + \bar{x}(\alpha_3 + \bar{x}(\alpha_4)))]$. ACX accumulates $\hat{f}(0) - f(0)$ into the print buffer. ADV prints what is in the buffer, right justified:

f(0) $\quad\quad$ $\hat{f}(0) - f(0)$ $\quad\quad$ in this case, with $\bar{x} = 0$..

Then the absolute value of $\hat{f} - f$ is taken before the program returns to LBL D.

LBL D then stores $|\hat{f}-f|$ in R6, squares this value and stores $|\hat{f}-f|^2$ in R7. The f index in R5 is incremented by 1 to prepare to retrieve f(1).

Next LBL8 is executed. The α index number, 15, is placed in the X stack position before execution jumps to LBL3.

LBL3 stores the α index, 15, into R3. Then FS?00 tests whether flag 0 is set, that is, whether the quartic term was dropped.

If the answer is no, that is, the quartic term was retained, the next line is skipped and the stack is rolled down to bring 1, corresponding to $\bar{x}=1$, into the X stack position and then the stacks are filled with $\bar{x}=1$. Guided by the α index in R3, which is 15 at this point, α_4 is indirectly recalled from R15. Multiplication yields 1 x α_4 or simply α_4. Then the α index in R3 is decremented by 1 to 14 before execution jumps to LBL12.

LBL12 indirectly recalls α_3, using the α index in R3. Since 1 is in both the T and Z stack positions, α_4 is in the Y position, and α_3 is in the X position, addition yields $\alpha_3 + \alpha_4$ and then multiplication by 1 yields the same. The α index is again decremented by 1 to 13 before the program returns to LBL3.

LBL3 immediately executes LBL12 again. This time α_2 is indirectly recalled by the α index in R3. With 1 in both the T and Z stack positions, $\alpha_3 + \alpha_4$ in the Y position, and α_2 in the X position, addition yields $\alpha_2 + \alpha_3 + \alpha_4$, and then multiplication by 1 yields the same. The α index is decremented by 1 to 12 before the program returns to LBL3.

In LBL3, FC?00 tests to see if flag 1 is cleared, that is, if the quartic term was retained.

If the answer is yes that the quartic term was retained, LBL12 is executed once again. This time α_1 is recalled indirectly by the α index. With 1 in both the T and Z stack positions, $\alpha_2 + \alpha_3 + \alpha_4$ in the Y position, and α_1 in the X position, addition yields $\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4$, and multiplication by 1 yields the same. The α index is decremented by 1 to 11 before the program returns to LBL3.

In LBL3, α_0 is indirectly recalled by the α index in R3. With $\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4$ in the Y stack position, addition yields $\alpha_0 + \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4$ which equals $\hat{f}(1)$. Then the program returns to LBL8 after the XEQ 3 command and jumps to LBL16.

In LBL16, f(1) is recalled indirectly by the f index in R5 and then accumulated in the print buffer. The next two steps rearrange the stacks:

T	1		1
Z	$\hat{f}(1)$	1. X<>T	3
Y	f(1)	2. RDN ⟶	$\hat{f}(1)$
X	3		f(1)

Subtraction yields $\hat{f}(1) - f(1)$, which is accumulated into the print buffer. Then f(1) and $\hat{f}(1) - f(1)$ are printed. The absolute value of $\hat{f}(1) - f(1)$ is taken before the program returns to LBL8.

In LBL8, $|\hat{f}(1) - f(1)|$ is added to the contents of R6 and the sum, $|\hat{f}(1) - f(1)| + |\hat{f}(0) - f(0)|$, is stored in R6. Then $|\hat{f}(1) - f(1)|^2$ is calculated and added to the contents of R7 and the sum, $|\hat{f}(1) - f(1)|^2 + |\hat{f}(0) - f(0)|^2$ is stored in R7. The stack is rolled up to bring 1 to the X stack position and 1 is added to yield $\bar{x}$=2. Then the f index in R5 is incremented by 1 to prepare to retrieve f(2) before the program goes to the beginning of LBL8.

LBL8 restores the α index to 15 before it jumps to LBL3.

LBL3 stores the α index of 15 in R3. Then it tests flag 0 again to see if the quartic term was dropped. If the answer is no, the next line is skipped again and the stacks are rolled down to bring $\bar{x}$=2 in the X position and then the stacks are filled with $\bar{x}$=2. Directed by the α index in R3, α_4 is indirectly recalled from R15 and multiplied by 2. Then the α index is decremented to 14 before LBL12 is executed.

LBL12 indirectly recalls α_3 from R14 by the α index. With 2 in both the T and Z stack positions, $2\alpha_4$ in the Y position, and α_3 in the X position, addition yields $\alpha_3 + 2(\alpha_4)$ and then multiplication yields $2(\alpha_3 + 2(\alpha_4))$. The α index is again decremented to 13 before the program returns to LBL3 which immediately executes LBL12 again.

This time, LBL12 indirectly recalls α_2 from R13 using the α index. With 2 in both the T and Z stack positions, $2(\alpha_3 + 2(\alpha_4))$ in the Y position, and α_2 in the X position, addition yields $\alpha_2 + 2(\alpha_3 + 2(\alpha_4))$ and then multiplication yields $2(\alpha_2 + 2(\alpha_3 + 2(\alpha_4)))$. The α index is decremented to 12 before the program returns to LBL3.

In LBL3, FC?00 tests to see if the quartic term was retained. If the answer is yes, LBL12, is executed once again.

This time, LBL12 indirectly recalls α_1, guided by the α index. With 2 in the T and Z stack positions, $2(\alpha_2 + 2(\alpha_3 + 2(\alpha_4)))$ in the Y position, and α_1 in the X position. Addition yields $\alpha_1 + 2(\alpha_2 + 2(\alpha_3 + 2(\alpha_4)))$ and then multiplication yields $2[\alpha_1 + 2(\alpha_2 + 2(\alpha_3 + 2(\alpha_4)))]$. The α index is decremented to 11 before the program returns to LBL3.

In LBL3, α_0 is indirectly recalled by the α index. With $2[\alpha_1 + 2(\alpha_2 + 2(\alpha_3 + 2(\alpha_4)))]$ in the Y stack position, addition yields $\alpha_0 + 2[\alpha_1 + 2(\alpha_2 + 2(\alpha_3 + 2(\alpha_4)))]$ which equals $\hat{f}(2)$. Then the program returns to LBL8, after the XEQ 3 command, to immediately execute LBL16.

LBL16 calculates $\hat{f}(2) - f(2)$ and $|\hat{f}(2) - f(2)|$ and prints f(2) and $\hat{f}(2) - f(2)$ in a manner analogous to the one previously described. Then the program returns to LBL8.

LBL8 adds $|\hat{f}(2) - f(2)|$ to the contents of R6 to yield $|\hat{f}(2) - f(2)| + |\hat{f}(1) - f(1)| + |\hat{f}(0) - f(0)|$ which is stored in R6. It then calculates $|\hat{f}(2) - f(2)|^2$ and adds this to the contents of R7 to yield $|\hat{f}(2) - f(2)|^2 +$

$|\hat{f}(1) - f(1)|^2 + |\hat{f}(0) - f(0)|^2$ which is stored in R7. Then the stack is rolled up to bring $\bar{x}$=2 to the X stack position and 1 is added to obtain $\bar{x}$=3. Finally, the f index in R5 is incremented by 1 to prepare to retrieve f(3).

At this point the program would return to the beginning of LBL 8 and the great loop of LBL8 to LBL3 to LBL12 to LBL16, and then back to LBL8 would be repeated. However, to address the matter of the flags if the quartic term were dropped, momentarily interrupt the quartic loop and return to LBL3 for the first test for flags, FS?00. If the answer is yes, that the quartic term had been dropped, the next step decrements the α index from 15 to 14 so that after $\bar{x}$=1 is filled in the stacks, α_3 is indirectly recalled from R14 by the α index in R3. Then the α index is decremented to 13 before LBL16 is executed and then decremented to 12 before LBL16 is executed again to yield $1[\alpha_1 + 1(\alpha_2 + 1(\alpha_3))]$. Then α is decremented to 11 before the program returns to LBL3 for the second test, FC?00. Since the answer to whether the quartic term was retained is no, the program skips the third execution of LBL12 and proceeds to indirectly recall α_0 from R11 using the α index. Then $\alpha_0 + 1[\alpha_1 + 1(\alpha_2 + 1(\alpha_3))]$ is calculated. Similarly, when the test FS?00 is encountered a second time with $\bar{x}$=2, the α index is decremented from 15 to 14 so that α_3 is recalled first and then $2[\alpha_1 + 2(\alpha_2 + 2(\alpha_3))]$ is eventually calculated. Also, when FC?00 is encountered again with $\bar{x}$=2, the third execution of LBL12 is akipped so that $\alpha_0 + 2[\alpha_1 + 2(\alpha_2 + 2(\alpha_3))]$ is then calculated.

Now, returning to the paragraph before the last one, and the position in the program after the steps ISG05 and GTO08 in LBL8, the great loop of LBL8, LBL3, LBL12, LBL16, and then LBL8 once more, is repeated again and again in exactly the same fashion as previously described. Each time 1 is incremented to the $\bar{x}$ value and also to the f index so that all the data is run through the calculations. At the end of the last execution of this loop, immediately preceeding the step, ISG05, in LBL8, $\Sigma|\hat{f}(x) - f(x)|$ will be stored in R6, $\Sigma|\hat{f}(x) - f(x)|^2$ will be stored in R7, and the printer read out will appear as:

$$\begin{array}{ll} f(0) & \hat{f}(0) - f(0) \\ f(1) & \hat{f}(1) - f(1) \\ \vdots & \vdots \\ f(N) & \hat{f}(N) - f(N) \end{array}$$

If FS?00 is true, these calculated values will not contain the quartic terms. Then since ISG 05 will not be true, that is, the f index will be exceeded after the last data point has been run through, the next step, GTO 08, will be skipped. The next set of calculations in LBL8 tabulate the average derivation and the standard derivation as follows:

$N + 1$ -- RCL00 + 1, in X stack position

$\Sigma|\hat{f}(\bar{x}) - f(x)|/(N+1)$ -- STO/06 : average derivation in R6

$N - 1$ -- $(N+1) - 2$

$\Sigma|\hat{f}(\bar{x}) - f(x)|^2/(N-1)$ -- STO/07

$\left(\frac{\Sigma|\hat{f}(\bar{x}) - f(x)|^2}{(N-1)}\right)^{1/2}$ -- RCL 7, SQRT : standard derivation

The standard derivation is printed and the average derivation is recalled from R6 and is also printed. Then the program returns to the normal mode.

LBL E converts the α_m values to a_m values. To execute LBL E, key in x_i, the initial value, press ENTER, key in I, the interval, and then key USER E. LBL E stores I in R1. The stack is rolled down to return x_i to the X stack position and then the stacks are filled with x_i. Next the FC?00 test checks to see if the quartic term was retained.

If the answer is yes that the quartic term was retained, the program jumps to LBL15 which calculates the contribution of α_4 to α_0 through a_4. The first series of steps places x_i in the T stack position, α_4 in R15 in the Z position, I in R1 in the Y position, and 4 in the X position. Taking y to the power of x yields I^4 and then dividing yields α_4/I^4 which is stored in R20. This is the contribution of α_4 to a_0 and also equals a_4. Thus, a_4 is stored in R20. With x_i now in the Y stack position, multiplication yields $\alpha_4 x_i/I^4$. Then the sign of this value is changed and it is multiplied by four to yield $-4\alpha_4 x_i/I^4$ which is the contribution of α_4 to a_1. This is stored in R19. With x_i again in the Y stack position, multiplication yields $-4\alpha_4 x_i/I^4$. Then multiplication by -1.5 yields $6\alpha_4 x_i^2/I^4$ which is the α_4 contribution to a_2. This is stored in R18. With x_i still in the Y stack position, multiplication yields $6\alpha_4 x_i^3/I^4$. Then division by -1.5 yields $-4\alpha_4 x_i^3/I^4$ which is the α_4 contribution to a_3. This is stored in R17. With x_i again in the Y stack position, multiplication yields $-4\alpha_4 x_i^4/I^4$. Then division by -4 yields $\alpha_4 x_i^4/I^4$ which is the α_4 contribution to a_4. This is stored in R16. Then the stack is rolled down to bring x_i to the X stack position before the program returns to LBLE after the XEQ 15 command. In LBL E, the test FS?00 is made. Since the quartic term was retained the answer to whether flag 0 was set is no, so that the next step, GTO 00, is skipped and the program continues with the step, RCL 14.

However, back to the first test, FC?00, in LBL E: if the answer had been no so that the quartic term had been dropped, the next step, XEQ 15, would be skipped and the test FS?00 would be encountered. This time the answer would be yes and LBL00 would be executed. LBL00 stores 0 in R16,17,18 and 19 since the contribution of α_4 to a_0 through a_3 does not exist if the quartic term is dropped. Thus, if at first the quartic term had been retained and then it was decided to drop it, LBL0 clears the registers in which the α_4 contribution to a_0 through a_3 had been stored. (It does not clear R20 which contains the α_4 contribution to a_4, or simply a_4, because this is not necessary.) Then the stack is rolled down to bring x_i to the X stack position. The program returns to LBL E after the XEQ 00 command and continues with the next step, RCL 14.

Hence, whether or not the quartic term was retained, the program continues by recalling α_3 in R14 and placing it in the Z stack position, placing I in R1 in the Y position, and 3 in the X position. Taking y to the power of x yields I^3 and then division yields α_3/I^3. Adding to the contents of R19 yields $\alpha_3/I^3 - 4\alpha_4 x_i/I^4$ which is stored in R19. This equals a_3. If the quartic term had been dropped the last term would be zero. Next, with x_i in the Y stack position and α_3/I^3 in the X position, multiplication yields $\alpha_3 x_i/I^3$ and then multiplication by 3 yields $3\alpha 3x_i/I^3$. Subtracting this term from the contents of R18 yields $-3\alpha 3x_i/I^3 + 6\alpha_4 x_i{}^2/I^4$ which is stored in R18. The last term is zero for a cubic fit. Then with x_i in the Y stack position and $3\alpha_3 x_i/I^3$ in the x position, multiplication yields $3\alpha_3 x_i^2/I^3$. This term is added to the contents of R17 to yield $3\alpha_3 x_i^2/I^3 - 4\alpha_4 x_i^3/I^4$ which is stored in R17. The last term is zero for a cubic fit. With x_i still in the Y stack position and $3\alpha_3 x_i^2/I^3$ in the X position, multiplication yields $3\alpha_3 x_i^3/I^3$ and then division by 3 yields $\alpha_3 x_i^2/I^3$. Subtracting this term from the contents of R16 yields $-\alpha_3 x_i^3/I^3 + \alpha_4 x_i^4/I^4$ which is stored in R16. The last term is again zero for a cubic fit. Then the stack is rolled down to keep x_i in the stack. The next series of steps places x_i in the Z stack position, α_2 in R13 in the Y position, and I in R1 in the X position. Squaring yields I^2 and then dividing yields α_2/I^2. This term is added to the contents of R18 which yields $-3\alpha_3 x_i{}^3/I^3 + \alpha_2/I^2 + 6\alpha_4 x_i^2/I^4$ which is stored in R18. This equals a_2. The last term is zero for a cubic fit. Then with x_i now in the Y stack position and α_2/I^2 in the X position, multiplication yields $\alpha_2 x_i/I^2$ and then multiplication by 2 yields $2\alpha_2 x_i/I^2$. This term is subtracted from the contents of R17 to yield $3\alpha_3 x_i^2/I^3 - 2\alpha_2 x_i/I^2 - 4\alpha_4 x_i^3/I^4$ which is then stored in R17. Again, the last term is zero for a cubic fit. With x_i again in the Y stack position and $2\alpha_2 x_i/I^2$ in the X stack position, multiplication yields $2\alpha_2 x_i^2/I^2$ and then division by 2 yields $\alpha_2 x_i^2/I^2$. This term is added to the contents of R16 to yield $-\alpha_3(x_i/I)^3 + \alpha_2(x_i/I)^2 + \alpha_4(x_i/I)^4$ which is stored in R16. The last term is zero for a cubic fit. Then the stack is rolled down to keep x_i in the stacks. The next series of steps places x_i in the Z stack position, α_1 in R12 in the Y position, and I in R1 in the X position. Division yields α_1/I. This term is added to the contents of R17 to yield $3\alpha_3 x_i^2/I^3 - 2\alpha_2 x_i/I^2 + \alpha_1/I - 4\alpha_4 x_i^3/I^4$. This equals a_1. The last term is zero for a cubic fit. Then with x_i in the Y stack position and α_1/I in the X position, multiplication yields $\alpha_1 x_i/I$. This term is subtracted from the contents of R16 to yield $-\alpha_3(x_i/I)^3 + \alpha_2(x_i/I)^2 - \alpha_1(x_i/I) + \alpha_4(x_i/I)^4$ which is stored in R16. Again the last term is zero for a cubic fit. Next, α_0 is recalled from R11 and added to the contents of R16 to yield $-\alpha_3(x_i/I)^3 + \alpha_2(x_i/I)^2 - \alpha_1(x_i/I) + \alpha_0 + \alpha_4(x_i/I)^4$ which is stored in R16. This equals a_0. Once again the last term is zero for a cubic fit. Finally, a_0 is recalled from R16 and the program stops to display a_0. Keying SST recalls a_1 from R17 and displays it. Keying SST again recalls a_2 from R18 and displays it, SST again recals a_3 from R19 and displays it, and if the quartic term was retained, keying SST a final time would recall a_4 from X20 and display it.

If it is desired to have a_0 to a_3 or a_4 yield dimensionless values of $-(G°-H°_{298})/RT$, at this point store R, the universal gas constant, in R90 and with the printer attached, key USER F. LBL F recalls R from R90 and then divides the a_m values in R16 through R20 by R to obtain a_0/R, a_1/R, a_2/R, a_3/R, and a_4/R if the quartic term was retained. The display is set to 5 figures beyond the decimal point in engineering notation in case the a_m/R values are very small. Entry of 16.02 in the X position is followed by FS?00.

If the test FS?00 is false, that is, the quartic term was retained, execution jumps to PRREGX and the command 16.02 (actually 16.020). PRREGX prints the contents of R16 through R20 which are a_0/R, a_1/R, a_2/R, a_3/R, and a_4/R. If the quartic term had been dropped, FS?00 is true and 16.02 in the X position is displaced by 16.019. The command 16.019 PRREGX tells the printer to print the contents of R16 though R19 which are a_0/R, a_1/R, a_2/R, and a_3/R.

LBL B calculates and displays individual $\hat{f}(\bar{x})$ values. To execute LBL B, enter a $\bar{x}$ value for which the $\hat{f}(\bar{x})$ value is desired; then press USER B. LBL B sets the α index to 15 before it goes to LBL 3. LBL 3 stores the α index, 15, in R3 and then fills the stacks with the $\bar{x}$ value entered. By decrementing the α index each time the loop is performed, the program goes through α_4 or α_3 to α_0, depending on whether the quartic term was retained or dropped. The program proceeds to calculate $\hat{f}(\bar{x}) = \alpha_0 + \bar{x}[\alpha_1 + \bar{x}(\alpha_2 + \bar{x}(\alpha_3 + \bar{x}(\alpha_4)))]$, where the last term would be zero for a cubic fit, in a manner analogous to the one previously described for LBL 3, going through LBL 12. After $\hat{f}(\bar{x})$ has been calculated, the program stops to display it.

LBL C calculates and displays individual $\hat{f}(x)$ values. To execute LBL C, enter a x value for which the $\hat{f}(x)$ value is desired; then press USER C. LBL C sets the a index to 20 before it moves to LBL 3. LBL 3 stores the a index, 20, in R3 and then fills the stacks with the x value entered. By decrementing the a index each time the loop is performed, the program runs through a_4 or a_3 to a_0, depending on whether the quartic term was retained or dropped. The program proceeds to calculate $\hat{f}(x) = a_0 + x[a_1 + x(a_2 + x(a_3 + x(a_4)))]$ in a procedure analogous to that described previously. After $\hat{f}(x)$ has been calculated, the program stops to display it. However, if USER F has been keyed already, the a registers would contain a_m/R so that $\hat{f}(x)/R$ would have been calculated instead. Since R is in R90 and LBL3 is followed by RCL90 * RTN, R/S will calculate $\frac{\hat{f}(x)}{R}$ R to obtain and display $\hat{f}(x)$.

Program CB can readily be modified to fit various types of data. For instance, if values of $-(G^\circ-H^\circ_{298})/T$ are to be fit into an analytic equation but the data points are given in reference to H°_0 so that values of $-(G^\circ-H^\circ_0)/T$ are listed instead; the following modification of Program CB can be made to convert values of $-(G^\circ-H^\circ_0)/T$ to $-(G^\circ-H^\circ_{298})/T$ before the data are fit to the Chebyshev polynomial equation.

To execute this modified version, labelled CBO, store R, the universal gas constant, in R90, the value of $H^\circ_{298}-H^\circ_0$ for the given substance in R91, the initial temperature, T_1, in R92, and the temperature interval between the data points, I, in R93. If the four quantities are put on the stack in order from R to I, LBLST STO93 RDN STO92 RDN STO91 RDN STO90 END will store the values with XEQ 'ST'. The maximum register numbers needed for storage, excluding R88 through R95, if $N \leq 16$ is:

$$20 + 4N \quad \text{for N odd} \quad \text{and} \quad 22 + 4N \quad \text{for N even.}$$

Two entirely new labels are introduced, LBL18 and LBL19. LBL 19 sets up the registers where $-(G^\circ-H^\circ_0)/T$ values will be stored and where $-(G^\circ-H^\circ_0)/T + (H^\circ_{298} - H^\circ_0)/T = -(G^\circ-H^\circ_{298})/T$ values will be stored. LBL 18 adds the $(H^\circ_{298} - H^\circ_0)/T$ values to the corresponding $-(G^\circ-H^\circ_0)/T$ values.

Upon entering the first data point, $f_0(0) = -(G^\circ-H^\circ_0)/T$, and then XEQ CBO, the program proceeds unchanged from the unmodified version until LBL11 is executed. LBL11 is changed by adding XEQ 19 and then XEQ 18 at the end of the original version. LBL19 consists of the following steps:

LBL19 RCL92 RCL93 - STO89 RDN RCL05 FRC 1E3 * 1.1 + STO94 RDN RTN

First, T_1 in R92 is reduced by I in R93 so that T_1 - I is stored in R89. The reason for this is that LBL18 increments the temperature, T, in R89, by I each time it is executed so that the first time the program executes LBL18, T will equal T_1. The the stack is rolled down to bring f(0) back to the X stack position, having been placed there before LBL19 had been executed. Next, the original f index, $q+1.\frac{q+1+N}{1000}$, is recalled from R5. The fractional part of the f index is taken and multiplied by 1000 to yield the register in which the last value of $-(G°-H°_{298})/T = f_{298}(N) = f_0(N) + (H°_{298} - H°_0)/T$ will be stored (by LBL10). Then 1.1 is added to yield a new index, $g_0 = q + 1 + N + 1.100$, for the input data, and this number is stored in R94. Thus, the original data will be stored beginning with R(q+N+2); 100 is merely a large enough counter test value to prevent skipping. Finally, the stack is rolled down to bring f(0) back to the X stack position before the program returns to LBL11 to immediately execute LBL18.

LBL18 consists of the following steps:

LBL18 STO IND94 ISG94 RCL93 ST+89 RDN RCL91 RCL89 / + RTN

First, $f_0(0)$ is indirectly stored in R(q+N+2) by the g_0 index in R94. Then this index is incremented by 1 in preparation for the next data point. I is recalled from R93 and added to the temperature, T, in R89 which had equalled T_1-I. The incremented temperature, T_1, is stored in R89. The stack is rolled down to return $f_0(0)$ to the X position.$(H°_{298} - H°_0)$ is recalled from R91 and divided by the temperature in R95, $T=T_1$, corresponding to $f_0(0)$. This value, $(H°_{298} - H°_0)/T_1$ is added to $f_0(0)$ before execution returns to LBL11 which then continues to LBL10.

No steps are altered in LBL10, but this time it indirectly stores the calculated $f_0(0) + (H°_{298} - H°_0)/T_1$ value, guided by the original f index, $q+1.\frac{q+1+N}{1000}$, in R5. Thus, the values for $f_{298}(\bar{x}) = f_0(\bar{x}) + (H°_{298}- H°_0)/T$, where T is continually incremented to correspond to its $f_0(\bar{x})$ value, are stored in R(q+1) through R(q+1+N).

Since LBL10 stores the $f_0(\bar{x}) + (H°_{298}- H°_0)/T$ values, LBL18 must be executed before each execution of LBL10. Thus, before each XEQ 10 in LBL 'CBO', LBL1, LBL2, and LBL5, an XEQ 18 must be inserted after steps 13, 29, 49, and 86, respectively. Then the modified program runs through the data in a similar manner to the original program, but with LBL18 storing $f_0(\bar{x})$ in R(q+N+2) through R(q+2N+2) using the g_0 index in R94 and storing $f_{298}(\bar{x})$ in R(q+1) through R(q+N+1) using the original f index in R5. Then the calculations for fit proceed in the same manner, using the $f_{298}(\bar{x})$ values.

The next modification of the program occurs in LBL E, as follows:

LBL E RCL93 STO01 RCL92 ENTER↑ . . . . .

Since the interval, I, is stored in R93 and the initial value, T_1, is stored in R92, these values are simply recalled rather than re-entered before keying USER E, as done with the original program.

The next alteration occurs in LBL D. An extra step, ST-94, should be inserted after step ST-05. Subtracting N+1 from the original f index in R5 had positioned the registers to the first $f_0(\bar{x})$ + $(H^\circ_{298} - H^\circ_0)/T$ value. Also subtracting N+1 from the g_0 index in R94 positions the registers to the first $f_0(\bar{x})$ value.

LBL16 which handles the printout is modified as follows:

LBL16 RCL IND94 ACX ISG94 RDN RCL IND05 ACX — ACX ADV ABS RTN

The first time through, $-(G^\circ - H^\circ_0)/T_1$ is indirectly recalled, directed by the g index in R94 which had been set to R(q+N+2) by LBL D before LBL16 was executed. This value is printed and the g index is incremented by 1 to prepare to print out the next $-(G^\circ - H^\circ_0)/T$ value. Then the stack is rolled down to bring $\hat{f}_{298}(0)$ = $-(G^\circ - H^\circ_{298})/T$ into the X position, having been placed in the stack previously by LBL D. Next, $f_{298}(0)$ = $-(G^\circ - H^\circ_{298})/T$ is indirectly recalled, guided by the f_0 index in R5, and this value is printed. Subtraction yields $\hat{f}_{298}(0)$ - $f_{298}(0)$, with H°_{298} as the reference state, and this value is also printed. LBL16 eventually runs through all the data as before, so that the printout is listed as follows:

$f_0(0) = -(G°-H°_0)/T_1$ $\hat{f}_{298}(0) = -(G°-H°_{298})/T_1$ $\hat{f}_{298}(0) - f_{298}(0)$

The final modification of Program CB occurs in LBL3. The following steps may be added after step 518, RTN:

RCL90 * RCL91 ↑ / -- RTN

These steps would be executed after USER C was keyed if USER F had been keyed earlier. Before these steps, $-(G°-H°_{298})/RT$ would be placed in the X stack position by LBL3, if LBL F had been previously executed. Therefore, R is recalled from R90 so that multiplication yields $-(G°-H°_{298})/T$. Then $H°_{298} - H°_0$ is recalled from R91. Since the value for T corresponding to $\hat{f}(x)$ had been keyed in along with USER C, the stack is rearranged as follows:

T	T		N+1
Z	T	R↑	$-(G°-H°_{298})/T$
Y	$-(G°-H°_{298})/T$	---------→	$H°_{298} - H°_0$
Y	$H°_{298} - H°_0$		T

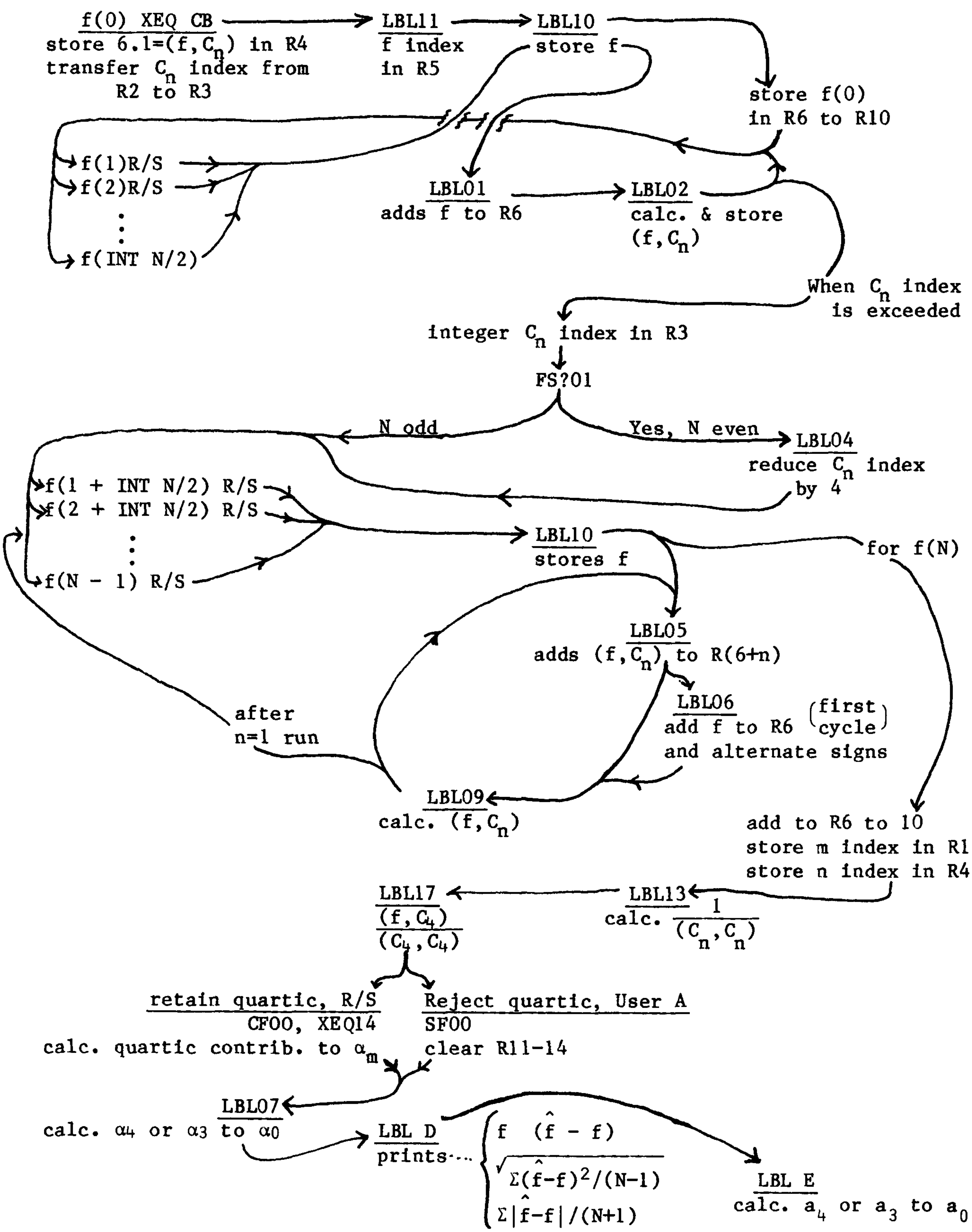
Therefore, division yields $(H°_{298} - H°_0)/T$ and then subtraction yields $-(G° - H°_{298})/T$ or $-(G° - H°_0)/T$, which is displayed.

The following change may be made if it is desired to recall the $f_0(\bar{x})$ values for a repeat fitting; for example if one of the values was incorrectly keyed in.

To repeat a fit, replace STO IND94 in LBL18 by RCL IND94. XEQ "CBO". Then key R/S for automatic retrieval of each value of $f_0(\bar{x})$.

With the foregoing modifications of Program CB, the program consists of 559 steps and 857 bytes.

Program CB


f(0) XEQ CB
store 6.1=(f,C_n) in R4
transfer C_n index from R2 to R3
LBL11
f index in R5
LBL10
store f
store f(0) in R6 to R10
f(1)R/S
f(2)R/S
f(INT N/2)
LBL01
adds f to R6
LBL02
calc. & store (f,C_n)
When C_n index is exceeded
integer C_n index in R3
FS?01
N odd
Yes, N even
LBL04
reduce C_n index by 4
f(1 + INT N/2) R/S
f(2 + INT N/2) R/S
f(N - 1) R/S
LBL10
stores f
for f(N)
LBL05
adds (f,C_n) to R(6+n)
LBL06
add f to R6 (first cycle)
and alternate signs
after n=1 run
LBL09
calc. (f,C_n)
add to R6 to 10
store m index in R1
store n index in R4
LBL17
$\frac{(f,C_4)}{(C_4,C_4)}$
LBL13
calc. $\frac{1}{(C_n,C_n)}$
retain quartic, R/S
CFOO, XEQ14
calc. quartic contrib. to α_m
Reject quartic, User A
SF00
clear R11-14
LBL07
calc. α_4 or α_3 to α_0
LBL D
prints....
f $(\hat{f} - f)$
$\sqrt{\Sigma(\hat{f}-f)^2/(N-1)}$
$\Sigma|\hat{f}-f|/(N+1)$
LBL E
calc. a_4 or a_3 to a_0

APPENDIX IIB (for Chapter II)

A number of supplementary programs are used for additional treatment of the analytical equations, $-(G°-H^°_{298})/RT = \Sigma a_n T^n$, obtained by programs CB or CBO. For example, it may be desired to round the values of a_n without changing the calculated values even at the highest temperature by more than the uncertainty or probable error, e, of the original data. Also, when fitting $-(G°-H^°_{298})/T$ values, it may be desired to have the calculated value at 298.15K fit exactly the value of $S^°_{298}$. Because the rounding error might happen to be in the same direction for most of the a_n values, a limiting rounding error, e/2, is applied to the contribution from each a_n value at the maximum temperature, T_{max}. The probable error is based on the uncertainty of $S^°_{298}$, which is usually the major source of uncertainty, and no account is taken of the increasing uncertainty due to error in the heat capacity values as the temperature is increased. When the uncertainty in $S^°_{298}/R$ is greater than 0.005, the value of e/R used in the rounding procedure is restricted to 0.005.

Program GG starts with the following quantities in registers 88 to 94, when used with program CBO:

R:	88	89	90	91	92	93	94
	e/R	T_{max}	R	$H^°_{298}- H^°_0$	T_1	I	g_0 index

Register 71 contains the index used for indirect storing of the final rounded and corrected a_n values in registers 72 to 87. The g_0 index in R94 is used for storing and retrieving $-(G°-H^°_0)/T$ values in registers q+N+2 to q+2N+2. As described in Appendix IIA, the f index in R5 deals with indirect storage and retrieval of values of $-(G°-H^°_{298})/T$ in registers q+1 to q+N+1. For example, for 13 data points ranging from 1000 to 3000K at intervals of I=200K, N = 12, q = 44, and the $-(G°-H^°_{298})/T$ values are stored in registers 45 to 57 and the $-(G°-H^°_0)/T$ values are stored in registers 58 to 70.

In addition to the a_n values in R16 to 19 or 20, the following additional quantities are utilized by program GG:

R:	1	3	4	5	6	7	8	9	10
	e/2R	a_n index	$10^m a_n$	f index	Δm	$e/2RT^n_{max}$	n_{max}	Δ	10^m

Operation of the rounding operation of program GG is outlined below. Flag 2 is set for m=6, 9, and 12. R8 contains $n_{max}=3$ if F00 is set or 4 if F00 is not set. $R7*R10 \rightarrow 10^m(e/2RT^n_{max})=0.h$ and FRC $10^m a_n=0.d$.

a_n R3 index	R6 Δm	LBL25 operation →	m	for $10^m a_n$ in R4 and 10^m in R10	
17.1	0	→	5	d>h to LBL26 d<h to LBL25	LBL24 $8 \not< 8$ to LBL25 on m=8 line
17.1	1	→	6	→ LBL26 → LBL21	
18.1	0	→	8	d>h to LBL26 d<h to LBL25	LBL24 8<11 to LBL20 lines a or b
18.1	1	→	9	→ LBL26 → LBL21	
19.1	0	→	11	d>h to LBL26 d<h to LBL25	LBL24 8<14 to LBL20 line c
19.1	1	→	12	→ LBL26 → LBL21	
20.1	0	→	14	d>h to LBL26 d<h to LBL25	LBL24 8<17 to LBL20 line c
20.1	1	→	15	→ LBL26 → LBL21	

LBL20 {
- a — to LBL H if F00 set, XEQ LBL25 on line m=11 above to round a_3
- b — to LBL25 on m=11 line above if $11 \not< m=11$
- c — to LBL I as 11<m=14 or 17, XEQ LBL25 on line m=14 above to round a_4

The rounding procedure starts with n=1, m=5, $\Delta m=0$ and 17.1 in R3. n_{max} is 3 if F00 is set and is 4 otherwise. LBL25 puts $10^5 a_1$ in R4 and $10^5 \times e/2RT_{max}$ yields the fraction 0.h which is compared with FRC $10^5 a_1=0.d$. If d<h, LBL26 rounds $10^5 a_1$ from R4, drops 0.d, divides by 10^5 and stores in R17. R3 is

incremented to 18.1, and m in R10 is increased to 8. LBL24 divides R7 by T_{max} from R89 and LBL25 commences the rounding of a_2. If d>h, Flag 2 is set, $\Delta m=1$ is put in R11, and m in R10 is increased to 6. Then LBL25 is repeated to find $10^6(e/2rT_{max})$ > FRC $10^6 a_1$. $10^6 a_1$ from R4 is then rounded by LBL26, divided by 10^6, and stored in R17. R3 is incremented to 18.1 and LBL21 increases m in R10 by $3-\Delta m=2$ to m=8 and changes Δm in R11 back to zero. Then LBL24 prepares for a_2 as indicated above.

When m in R10 has been increased to 11, LBL20 checks Flag00. If F00 is not set ($n_{max}=4$ in R8), LBL25 commences rounding of a_3. If it is desired to round the last a_n value, XEQ 25 will round it and stop again at LBL H or LBL I. R/S with printer attached will then print calculated values of $-(G°-H_0°)/T$ at T_1 and T_{max}, and rounded values of a_n.

When T_1 is 300K and it is desired to have the calculated value at 298.15K fit exactly the value of $S°_{298}$, the first and last a_n values are modified to increase the calculated value $\hat{y}$ at 298 or 300K by $\Delta=y-\hat{y}$ to provide an exact fit and to reduce the calculated value at T_{max} by Δ so that the fit at high temperature is not changed by the adjustment at 298K. The two equations,

$$y-\hat{y} = \Delta = \Delta a_0+\Delta a_n(298.15)^n \quad \text{and} \quad 0 = \Delta a_0+\Delta a_n(T_{max})^n,$$

yield: $$\Delta a_n = \Delta/(298.15^n-T^n_{max}) \quad \text{and} \quad \Delta a_0 = \Delta-(298.15)^n\Delta a_n.$$

If this procedure is to be followed, program GG is carried out to round the intermediate a_n values until LBL H or LBL I is reached. If R/S is then keyed, a_0 and the last a_n are modified to provide the exact fit at 298K and both are rounded. R/S will print out the calculated values at 298.15 and T_{max} based on the revised and rounded values followed by a printout of the final a_n values.

It is often desired to not only print out the constants along with the name of the species, the temperature range, and uncertainty, but it is convenient to store the information in the main storage registers of the calculator, or in the extended memory, or in a cassette to allow ready retrieval of the constants for use in calculations without having to key them in.

Program P can be run after CH and before CB or CBO to store the name and state of the species and the temperature range of fit in registers starting with R72. As each register holds only six characters and the program rotates the entries in the Alpha register six at a time, sufficient spaces should be included to yield a total of 18 characters or, if Flag 4 is set, a total of 24 characters. The entire line will be printed out. Before initiating program P for the first time, the index value 72.1 should be stored in R71. If program CBO is being used, only registers 72 to 87 are available. When the available registers have been used, the stored data is transferred to extended memory or to a cassette and 72.1 is stored again in R71 for a new set of entries. The storage in the Alpha register is simplified by use of ARCL to abtain the following register contents:

R:	95	96	97	98	99
	K,e/R=	<S,L>	0-1000	0-2900	0-3000

The procedure will be illustrated first for 0_2 gas. The quotation marks indicate entries in ALPHA mode.

72.1 STO71, '02<G>100 ARCL99 Sp K Sp' XEQ 'P'.

The printout is 02<G>1000-3000 K and 02<G>1 is stored in R72, 000-30 is stored in R73, 00 K is stored in R74.

The inclusion of uncertainty will be illustrated for O gas.

72.1 STO71 'O<G>30 ARCL97 ARCL95' XEQ 'P' will print out O<G>300-1000K,e/R=, and if this is now followed by 0.002 R/S, the 0.002 will be printed out on the second line. O<G>30 will be stored in R72, 0-1000 will be stored in R73, K,e/R= will be stored in R74, and 0.002 will be stored in R75 and in R88 where it will be used subsequently by program G for the rounding operations. The first six characters stored in the first register should contain enough information to identify the species, its state, and the temperature range if equations are given for two temperature ranges for the same state, as the contents of the first register will also be used as a data file name for storage in extended memory or in a cassette. Seven examples are given below to indicate the entries in the ALPHA mode and the printout. The printout is separated into the contents of each storage register. The actual printout will not have any gaps. For MgO and Al_2O_3, a bracket was omitted so that the temperature range beginning at 300K could be distinguished in the file name from the range beginning at 1000K for MgO or the label for Al_2O_3 solid could be distinguished from the label for Al_2O_3 liquid. For O_2 and MgO, the equals sign was deleted to save a register as e/R=0.002 or e/R 0.002 are equivalent.

	R72	R73	R74	R75 SF4
'0<G>30 ARCL97 ARCL95'	0<G>30	0-1000	K,e/R=	
'02<G>30 ARCL97 ARCL95 CLA'	02<G>3	00-100	0K,e/R	
'AL ARCL96 ARCL97 ARCL95 SpSpSp'	AL<S,L	>300-1	000K,e	/R=
'MGO<G30 ARCL97 ARCL95 CIA'	MGO<G3	00-100	0K,e/R	
'AL203S>30 ARCL97 ARCL95 SpSpSp'	AL203S	>300-1	000K,e	/R=
'MO8023<S>30 ARCL97 ARCL95 Sp'	MO8023	<S>300	-1000K	,e/R=
'TI305<a,b>30 ARCL97 ARCL95'	TI305<	a,b>30	0-1000	K,e/R=

The index for storing in the registers is automatically incremented, but when the registers have been used up and the stored data transferred to extended memory or to a cassette the next entries must be preceeded by 72.1 STO71.

After entry of the characters in the ALPHA register, keying 0.002, for example, followed by R/S will print 0.002 on the second line. However, if a number greater than 0.005, e.g. 0.01, is keyed in, the printout will read 0.01, USE 0.005 and 0.005 will be stored in R88 in place of 0.01 for the rounding operations of program GG. However, the e/R value of 0.01 will be stored in the register following the register which contains e/R.

After the use of program P, the data are entered as described in Chapter II and Appendix IIA followed by the rounding operations of program G described above in this appendix.

There are other auxillary programs that are convenient to use with program CBO. Program CBO requires the storage of R, $H^\circ_{298} - H^\circ_0$, T_1, and I in registers 90 to 93. The entry of these four values followed by XEQ ST will store them in the proper registers, as noted in Appendix IIA. For subsequent calculations, if only $H^\circ_{298} - H^\circ_0$ needs to be changed, it can be stored in R91 without using program ST.

The above discussion of program P described the storage of information about the species, its state, temperature range covered, and the e/R uncertainty. Following the rounding of the a_n values, XEQ SR will shift the a_n values from registers 16 to 19 or 20 and store them in the registers following the register containing the e/R value as directed by the index in R71. After all the available registers have been used, the data can be stored in extended memory or in a cassette using programs REGE or REGC. Two sets of information are needed for these programs. First the data file name is required in the ALPHA register which can be provided by the name in the first register filled in program P, e.g. ARCL72 or perhaps ARCL80, and the total number of registers in the X register. In the example of O gas with a quartic fit between 300 and 100K, nine registers would be required. If only a cubic fit were selected, eight registers would be required. For Ti_3O_5 between 300 and 1000K with a quartic fit, ten registers would be required. With those two entries, XEQ REGE will prepare a file in extended memory. Then the numbers of the registers to be moved must be inserted in the X register followed by R/S. For the example of Ti_3O_5 with ten registers starting in R72, the entry would be 72.081 R/S. Exactly the same procedure is used to store in a cassette. For the Ti_3O_5 example, the steps would be 'ARCL72' 10 EXQ'REGC' followed by 72.081 R/S.

To retrieve the Ti_3O_5 data from extended memory, one would key in 'TI305<' XEQ'EREG' followed by bbb.eee R/S where the data are to be stored in registers bbb to eee. To retrieve the Ti_3O_5 data from the cassette, one would key in 'TI305<' XEQ'CREG' followed by bbb.eee R/S.

If one were fitting N+1 tabulated values of $g_0=-(G°-H_0^°)/T$ at regular intervals using program CBO together with the auxillary programs described above, the sequence of steps would be as follows:

(1) N+1 XEQ'CH', R/S

(2) Enter in ALPHA register the name of species, state, and temperature range, using ARCL 96 to 99 as appropriate, followed by ARCL 95, for a total of 18 or 24 characters, spaces and commas. SF4 if 24 characters. Tone will sound when 24 characters have been entered. 72.1 STO71 if initiating storage.

(3) Attach printer in MAN mode and XEQ 'P'.

(4) Value of e/R R/S.

(5) $R\uparrow H_{298}^° - H_0^° \uparrow T_1 \uparrow I$ XEQ 'ST' unless values of R, T_1 and I are unchanged from previous fit; then $H_{298}^°-H_0^°$ STO91.

(6) $g_0(0)$ XEQ'CBO' → $g_{298}(0)$
$g_0(1)$ R/S → $g_{298}(1)$
⋮ ⋮
$g_0(N-1)$ R/S → $-g_{298}(N-1)$.
$g_0(N)$ R/S → e_{quar}, error due to dropping quartic term.

(7) If quartic term selected, R/S with printer in MAN mode.

If cubic fit selected, User A with printer in MAN mode.

As T_1 and I have been stored in step(5), it is not necessary to stop and initiate D, E, F, and GG as in program CB. As a result of initiating step 7, two printouts will take place. The first will print all the values of g_0 inserted, the resulting values of g_{298}, and the difference $\hat{g}_{298}-g_{298}$ between the values calculated from the analytical equation and the value obtained from the entered values, followed by the standard error and the average deviation. The second printout will give the a_n values of $-(G°-H°_{298})/RT = \Sigma a_n T^n$. For a cubic fit, 8 will be displayed in the X register. For a quartic fit, 11 will be displayed.

(8a) If T_1 = 300K and it is desired to fit S_{298}/R exactly, initiate the modification by R/S. 8 or 11 will be displayed again. R/S with MAN printer will print out the calculated values of $-(G°-H°_{298})/T$ at 298.15K and T_{max} based on the final a_n values which are also printed.

(8b) If T_1 is not 300K or it is not necessary to fit S_{298}/R exactly, XEQ 25 will round the last a_n value. RCL16 FIX3 RND STO16 will round a_0. R/S will print out calculated g_0 values at the extreme temperatures and the values of a_n.

(9) Either step (8a) or (8b) will also initiate program SR which transfers the final a_n values to the register following the register containing the e/R value as directed by the index in R71.

It is often convenient to fit data for the 298-1000K and 1000K-3000K separately but to store the constants together. If the lower temperature data are fit first and the name and constants stored starting with register 72, the index in R71 will store the constants for the higher temperature in the registers following those used for the first set. It is not necessary to repeat the uncertainty in the second set. The versions of programs SR and REGE given below will retrieve R72 to use as a file name and will calculate the number of registers used from the R71 index. With that version, no additional data have to be inserted after the printout of the rounded constants. Keying R/S will initiate program REGE and automatically transfer all the stored information to extended memory.

The steps listed below for programs P, ST, CBO, GG, SR, REGE, and EREG are given with GG and CBO combined in a single program. They are stored in extended memory or on magnetic tapes separately to allow GG to be used either with CB or CBO. However, once they are recalled, END at the end of CBO is deleted to allow automatic initiation of GG after completion of the subroutine of LBL F and to allow subroutine 22 of GG to utilize LBL C of program CB or CBO. Program CH, which is given in Chapter 2, is normally used with deletion of subroutines B, C, and 2 to reduce the number of steps to 105. With the 35 steps of LBL P, the 9 steps of LBL ST, the 684 steps of CBO combined with GG, and the 39 steps of the combined SR, REGE, and EREG programs,there is a total of 872 steps. The number of bytes is 1395, corresponding to use of 199 registers for the programs.

```
 01♦LBL "P"
ACA  XEQ 00  XEQ 00
XEQ 00  FS? 04  XEQ 00
PRBUF  RTN  STO 88
FIX 4  STO IND 71
ISG 71  ACX  .005  X<Y?
GTO 01  X<>Y  PRBUF  RTN

 21♦LBL 00
ASTO IND 71  ISG 71  6
AROT  RTN

 27♦LBL 01
STO 88  AON  ",USE"
AOFF  ACA  ACX  PRBUF
END
```

PRP "ST"

```
 01♦LBL "ST"
STO 93  RDN  STO 92  RDN
STO 91  RDN  STO 90  END
```

PRP "CB0"

```
 01♦LBL "CB0"
6.1  STO 04  RCL 02
STO 03  RCL Z  XEQ 11
STO 06  STO 07  STO 08
STO 09  STO 10  RTN
XEQ 18  XEQ 10

 16♦LBL 01
ENTER↑  ENTER↑  ENTER↑
ST+ IND 04  ISG 04
XEQ 02  XEQ 02  XEQ 02
XEQ 02  5  ST- 04  R↑
RTN  XEQ 18  XEQ 10
GTO 01

 33♦LBL 02
RDN  RCL IND 03  *
ST+ IND 04  ISG 04
ISG 03  RTN  RDN  RCL 03
INT  STO 03  RDN  DSE 03
FS? 01  XEQ 04  STOP
XEQ 18  XEQ 10  GTO 05

 53♦LBL 18
STO IND 94  ISG 94
RCL 93  ST+ 89  RDN
RCL 91  RCL 89  /  +
RTN

 64♦LBL 04
4  ST- 03  RDN  RTN

 69♦LBL 05
ENTER↑  ENTER↑  XEQ 09
ST+ 10  RDN  XEQ 06  RDN
XEQ 09  ST+ 09  RDN
XEQ 09  ST+ 08  RDN
XEQ 09  ST+ 07  R↑  STOP
XEQ 18  XEQ 10  GTO 05

 90♦LBL 09
RCL IND 03  DSE 03  *
RTN

 95♦LBL 06
STO T  STO Y  CHS  STO Z
RDN  ST+ 06  RTN

103♦LBL A
STO 20  0  STO 11
STO 12  STO 13  STO 14
SF 00  GTO 07

112♦LBL 19
RCL 92  RCL 93  -
STO 89  RDN  RCL 05  FRC
1 E3  *  1.1  +  STO 94
RDN  RTN

127♦LBL 11
RCL 02  FRC  1 E3  *  1
+  STO 05  RCL 00  +
1 E3  /  ST+ 05  RDN
XEQ 19  XEQ 18

143♦LBL 10
STO IND 05  ISG 05  RTN
XEQ 06  ST+ 10  RDN
ST+ 09  RDN  ST+ 08  RDN
ST+ 07

155♦LBL 17
0  STO 01  4  STO 04
XEQ 13  RCL 10  *
STO 11  STOP  CF 00  1
XEQ 14  STO 20  1
STO 01  RDN  XEQ 14
STO 19  ST- 20  2
STO 01  RDN  XEQ 14
STO 18  2  *  ST+ 20  3
STO 01  RCL 18  XEQ 14
RCL 20  X<>Y  STO 20  6
*  -  RCL 11  *  STO 12
RCL 19  RCL 18  3  *  -
RCL 20  11  *  +  RCL 11
*  STO 13  RCL 18
RCL 20  6  *  -  RCL 11
*  STO 14  RCL 20
RCL 11  *  STO 15

220♦LBL 07
0  STO 01  3  STO 04  1
XEQ 14  STO 19  1
STO 01  RDN  XEQ 14
ST- 19  STO 17  2
STO 01  RDN  XEQ 14
STO 18  2  *  RCL 19  +
XEQ 13  RCL 09  *
STO 19  ST+ 11  *
ST+ 12  RCL 17  RCL 18
3  *  -  RCL 19  *
ST+ 13  RCL 18  RCL 19
*  ST+ 14  DSE 04  0
STO 01  1  XEQ 14
STO 19  1  STO 01  RDN
XEQ 14  STO 17  ST- 19
XEQ 13  RCL 08  *
ST+ 11  STO 18  RCL 19
*  ST+ 12  RCL 17
RCL 18  *  ST+ 13
DSE 04  0  STO 01  1
XEQ 14  XEQ 13  RCL 07
*  ST+ 11  *  ST+ 12
RCL 06  RCL 00  1  +  /
ST+ 11  GTO D

304♦LBL 14
RCL 01  1  +  X↑2  LASTX
RCL 04  +  /  1/X
RCL 01  RCL 04  -  *
RCL 00  RCL 01  -  /  *
RTN

324♦LBL 13
RCL 00  FACT  X↑2
RCL 00  RCL 04  -  FACT
/  RCL 00  RCL 04  +  1
+  FACT  /  RCL 04  2  *
1  +  *  RTN

347♦LBL E
RCL 93  STO 01  RCL 92
ENTER↑  ENTER↑  ENTER↑
FC? 00  XEQ 15  FS? 00
XEQ 00  RCL 14  RCL 01
3  Y↑X  /  ST+ 19  *  3
*  ST- 18  *  ST+ 17  *
3  /  ST- 16  RDN
RCL 13  RCL 01  X↑2  /
ST+ 18  *  2  *  ST- 17
*  2  /  ST+ 16  RDN
RCL 12  RCL 01  /
ST+ 17  *  ST- 16
RCL 11  ST+ 16  GTO F
```

```
398♦LBL 15
RCL 15  RCL 01  4  Y↑X
/  STO 20  *  CHS  4  *
STO 19  *  CHS  1.5  *
STO 18  *  CHS  1.5  /
STO 17  *  CHS  4  /
STO 16  RDN  RTN

427♦LBL 00
0  STO 16  STO 17
STO 18  STO 19  RDN  RTN

435♦LBL 16
RCL IND 94  ACX  ISG 94
RDN  RCL IND 05  ACX  -
ACX  ADV  ABS  RTN

447♦LBL D
FIX 3  RCL 00  1  +
ST- 05  ST- 94  RCL 11
XEQ 16  STO 06  X↑2
STO 07  1  ST+ 05

461♦LBL 08
15  XEQ 03  XEQ 16
ST+ 06  X↑2  ST+ 07  R↑
1  +  ISG 05  GTO 08
RCL 00  1  +  ST/ 06  2
-  ST/ 07  RCL 07  SQRT
PRX  RCL 06  PRX  GTO E

486♦LBL B
15  GTO 03

489♦LBL C
20

491♦LBL 03
STO 03  FS? 00  DSE 03
RDN  ENTER↑  ENTER↑
ENTER↑  RCL IND 03  *
DSE 03  XEQ 12  XEQ 12
FC? 00  XEQ 12
RCL IND 03  +  RTN

509♦LBL 12
RCL IND 03  +  *  DSE 03
RTN
```

```
515♦LBL F
RCL 90  ST/ 16  ST/ 17
ST/ 18  ST/ 19  ST/ 20
ENG 5  16.02  FS? 00
16.019  PRREGX

527♦LBL "GG"
RCL 88  2  /  STO 01
RCL 89  /  STO 07  1 E5
STO 10  17.1  STO 03  0
STO 06  3  STO 08
FS? 00  GTO 25  1
ST+ 08

547♦LBL 25
RCL 07  RCL 10  *
RCL IND 03  RCL 10  *
STO 04  FRC  ABS  X<Y?
GTO 26  SF 02  1  ST+ 06
10  ST* 10  GTO 25

565♦LBL 26
RCL 04  FIX 0  RND
RCL 10  /  STO IND 03
ISG 03  FS? 02  GTO 21
1 E3  ST* 10

577♦LBL 24
RCL 89  ST/ 07  RCL 10
LOG  8  X<Y?  GTO 20
GTO 25

586♦LBL 20
FS? 00  GTO H  RDN  11
X<Y?  GTO I  GTO 25

594♦LBL 21
3  RCL 06  -  10↑X
ST* 10  CF 02  0  STO 06
GTO 24

604♦LBL H
STOP  16.019  STO 04
RCL 03  20  X<=Y?
GTO 23  GTO 27

613♦LBL I
STOP  16.020  STO 04
RCL 03  21  X<=Y?
GTO 23
```

```
621♦LBL 27
RCL 00  1  +  ST- 05
RCL IND 05  RCL 90  /
STO 04  298.15  STO 92
XEQ C  RCL 04  -  CHS
STO 09  RCL 92  RCL 08
Y↑X  RCL 09  RCL 08  Y↑X
-  /  ST+ 20  RCL 92
RCL 08  Y↑X  *  RCL 09
-  ST- 16  RCL 16  FIX 3
RND  STO 16  RCL 08  .1
+  ST+ 03  GTO 25

662♦LBL 23
FIX 3  RCL 92  XEQ 22
RCL 89  XEQ 22  RCL 04
ENG 5  PRREGX  GTO "SR"

672♦LBL 22
ACX  2  SKPCHR  RDN
XEQ C  RCL 90  *  ACX
ADV  END
                PRP "SR"

 01♦LBL "SR"
RCL 71  INT  STO 70
1 E3  /  16  +  RCL 08
ST+ 70  1  +  ST+ 71
1 E6  /  +  REGMOVE  RTN

 19♦LBL "REGE"
CLA  ARCL 72  RCL 70  71
-  CRFLD  RCL 70  1 E3
/  72  +  SAVERX  RTN

 33♦LBL "EREG"
0  SEEKPTA  RTN  GETRX
END
```

APPENDIX III (for Chapter III)

The combined programs ab1, ab2, abc2, and abc3 carry out the same calculations in the first 36 steps except for the flag setting of F1 if ab1 is initiated, F2 if ab2 is initiated, F3 if abc2 is initiated, and no specific flag if abc3 is initiated and the clearing of R19 for abc2 and abc3, and the clearing of R17 and R18 for all but ab1. For all programs, registers 0, 1, 3, 4 and 11-16 are cleared and 22.1 is stored in R20 as an index for storage of the x,y input data. F0 is also cleared.

In response to the queries from the calculator after the first step, F0 is set if $w \neq 1$, and if the x values are at regular intervals of I, I is stored in R0.

<u>Data Entry</u>

After the initial step, the data are inserted followed by User mode E in four possible manners as indicated in the directions depending upon whether the weighting factor w is 1 or not and whether the x values come at intervals of I or not. The treatment of the inserted values is indicated by the following outline.

LBLE:
- SF0($w \neq 1$) → LB7 Add w_i to R16, $\sqrt{w_i}$ to R8, decrement R16 so only $w_i - 1$ is added, remove 1 and $\sqrt{w_i}$ from stack → Go to LBL8
- CF0 → Store 1 in R8, remove 1 → Go to LBL8

LBL8: Store y_i in IND20, increment R20, calc $f(y_i)$ by LBL1, store $\sqrt{w_i}\ f(y_i)$ in R10,
store x_i in IND20, increment R20, use LBL3 to store x_i in R6, calc x_i' and store in R5, calc $f_1(x_i')$ and $f_1\sqrt{w_i}$ and store in R9.
Then branch to specific calculations for each program.

- SF1 → LBL9
- CF1 → Calc $f_2(x_i')$ by LBL4; then $f_2\sqrt{w_i}$ in R7
  - SF2 → LBL11
  - CF2
    - SF3 → LBL12
    - CF3 → LBL15

For all program upon completion of entry of each (x_i, y_i, w_i), x_i from R6 plus I from R0 displays $x_i + I$ for next data entry. In the following summaries, y will be used for $f(y_i)$, f_1 for $f_1(x_i')$, f_2 for $f_2(x_i')$, etc. and subscript i is dropped on w, x, and y. n will be used for the total number of x,y sets inserted, and $\Sigma w = \sum_{i}^{n} w_i$. A storage entry for a register which is not used in the specific program is indicated by --- on pg. 24.

ab1 LBL9: $(\sqrt{w} - 1)f\sqrt{w} = fw - f\sqrt{w}$ added to R11
$(\sqrt{w} - 1)y\sqrt{w} = yw - y\sqrt{w}$ added to R13
$y\sqrt{w}$ and $f\sqrt{w}$ are processed by Σ+ to add $f\sqrt{w}$ to previous $fw - f\sqrt{w}$ in R11 to yield net addition of fw and similar calculation for yw. w - 1 had been previously added to R16 so net addition is w. Additions to R12, 13, and 15 are f^2w, yw, and yfw, respectively.

ab2 LBL11: $f_2\sqrt{w}$ times $y\sqrt{w}$ from R10 is added to R18. From R9, $f_1\sqrt{w}$ times $y\sqrt{w}$ from R10 is added to R17. Then $f_2\sqrt{w}$ and $f_1\sqrt{w}$ are processed by Σ+. Only Σf_1^2w in R12, f_2^2w in R14, and f_1f_2w in R15 are used.

abc2 LBL12: $f_2\sqrt{w}$, $\sqrt{w}$ from R8 minus 1 STO3, $(f_2\sqrt{w})(\sqrt{w} - 1) = f_2w - f_2\sqrt{w}$ ST+13, $f_1w - f_1\sqrt{w}$ ST+11, $(y\sqrt{w})\sqrt{w} = yw$ ST+19, $f_2\sqrt{w}$ to LBL11 as for ab2 above.

abc3 LBL15: $f_2\sqrt{w}$ times $y\sqrt{w}$ from R10 adds yf_2w to R12. $(R9)^2 = f_1^2w$ ST+14. $(R7)^2 = f_2^2w$ ST+17. R9R7 = f_1f_2w ST+15. R10R9 = yf_1w ST+11. LBL5 calculates f_3. $f_3\sqrt{w}$ fills the stack. f_3^2w ST+19. yf_3w ST+13. $f_1f_3w - w + 1$ ST+16 (-w+1 compensates for LBL7). f_2f_3w ST+18.

Least-Squares Calculations

The next section outlines the calculations after all of the data have been entered. For each program, the least-square equations are derived and the calculations given in terms of storage registers used.

ab1 To minimize $[\Sigma(y-a-bf)]^2$, differentiation with respect to a and then b and setting differentials equal to zero yields

$a\Sigma w + b\Sigma fw = \Sigma yw$ $\qquad b = \dfrac{\Sigma yfw - (\Sigma yw\Sigma fw)/\Sigma w}{\Sigma f^2w - (\Sigma fw)^2/\Sigma w}$

$a\Sigma fw + b\Sigma f^2w = \Sigma yfw$ $\qquad a = (\Sigma yw - b\Sigma fw)/\Sigma w$

After insertion of all data, the remainder of LBL9 calculates

$\ell = R15 - R11R13/R16 = \Sigma yfw - \Sigma fw\Sigma yw/\Sigma w$

$m = R12 - (R11)^2/R16 = \Sigma f^2w - (\Sigma fw)^2/\Sigma w$

$b = \ell/m$ is stored in R2.

MEAN places $\Sigma fw/\Sigma w$ in x register and $\Sigma yw/\Sigma w$ in y register.

$a = \Sigma yw/\Sigma w - b\Sigma fw/\Sigma w$ is stored in R1.

ab2 Minimization of $[\Sigma(y-af_1-bf_2)]^2$ yields

$a\Sigma f_1^2w + b\Sigma f_1f_2w = \Sigma yf_1w$ $\qquad b = \dfrac{\Sigma f_1^2w\Sigma yf_2w - \Sigma yf_1w\Sigma f_1f_2w}{\Sigma f_1^2w\Sigma f_2^2w - (\Sigma f_1f_2w)^2}$

$a\Sigma f_1f_2w + b\Sigma f_2^2w = \Sigma yf_2w$ $\qquad a = [\Sigma yf_1w - b\Sigma f_1f_2w]/\Sigma f_1^2w$

The remainder of LBL11 calculates

$j = R12R18 - R17R15 = \Sigma f_1^2w\Sigma yf_2w - \Sigma yf_1w\Sigma f_1f_2w$

$k = R12R14 - (R15)^2 = \Sigma f_1^2w\Sigma f_2^2w - (\Sigma f_1f_2w)^2$

$b = j/k$ is stored in R3. $a = (R17 - bR15)/R12 = (\Sigma yf_1w - b\Sigma f_1f_2w)/\Sigma f_1^2w$ is stored in R2.

abc2 Minimization of $[\Sigma(y-a-bf_1-cf_2)]^2$ yields

$a\Sigma w + b\Sigma f_1w + c\Sigma f_2w = \Sigma yw$

$a\Sigma f_1w + b\Sigma f_1^2w + c\Sigma f_1f_2w = \Sigma yf_1w$

$a\Sigma f_2w + b\Sigma f_1f_2w + c\Sigma f_2^2w = \Sigma yf_2w$

The remainder of LBL12 calculates $R19/R16 = \Sigma yw/\Sigma w$, stored in R10 and

$q = (R11)^2/R16 - R12 = (\Sigma f_1w)^2/\Sigma w - \Sigma f_1^2w$ stored in R3.

$s = R11R13/R16 - R15 = \Sigma f_1w\Sigma f_2w/\Sigma w - \Sigma f_1f_2w$ stored in R6.

$r = (R13)^2/R16 - R14 = (\Sigma f_2w)^2/\Sigma w - \Sigma f_2^2w$ stored in R5.

$u = R10R13 - R18 = \Sigma yw\Sigma f_2w/\Sigma w - \Sigma yf_2w$ stored in R8.

$t = R10R11 - R17 = \Sigma yw\Sigma f_1w/\Sigma w - \Sigma yf_1w$ stored in R7.

$$b = \frac{R5R7 - R6R8}{R3R5 - (R5)^2} = \frac{rt - su}{qr - s^2} \text{ stored in R2.}$$

$$c = \frac{R7 - R2R3}{R6} = \frac{t - bq}{s} \text{ stored in R3.}$$

$a = R10 - [R3R13 + R11R2]/R16 = \Sigma yw/\Sigma w - [c\Sigma f_2w + b\Sigma f_1w]/\Sigma w$
stored in R1.

abc3 Minimization of $[\Sigma(y-af_1-bf_2-cf_3)]^2$ yields

$$a\Sigma f_1w + b\Sigma f_1f_2w + c\Sigma f_1f_3w = \Sigma yf_1w$$
$$a\Sigma f_2w + b\Sigma f_2^2w + c\Sigma f_2f_3w = \Sigma yf_2w$$
$$a\Sigma f_3w + b\Sigma f_2f_3w + c\Sigma f_3^2 w = \Sigma yf_3w$$

The remainder of LBL 15 calculates

$A = R17R19 - (R18)^2 = \Sigma f_2^2w\Sigma f_3^2w - (\Sigma f_2f_3w)^2$, stored in R4,

$B = R15R19 - R16R18 = \Sigma f_1f_2w\Sigma f_3^2w - \Sigma f_1f_3w\Sigma f_2f_3w$, stored in R5,

$C = R15R18 - R16R17 = \Sigma f_1f_2w\Sigma f_2f_3w - \Sigma f_1f_3w\Sigma f_2^2 w$, stored in R6,

$D = C(R16) + R14R4 - R15R5 = C\Sigma f_1f_3w + A\Sigma f_1^2w - B\Sigma f_1f_2w$ in R7,

$R = R12R18 - R13R17 = \Sigma yf_2w\Sigma f_2f_3w - \Sigma yf_3w\Sigma f_2^2w$ in R8,

$Q = R12R19 - R13R18 = \Sigma yf_2w\Sigma f_3^2w - \Sigma yf_3w\Sigma f_2f_3w$ in R9,

$a = [R(R16)-Q(R15)+R4R11]/R7 = (R\Sigma f_1f_3w - Q\Sigma f_1f_2w + A\Sigma yf_1w)/D$ in R2,

$S = R13R15 - R12R16 = \Sigma yf_3w\Sigma f_1f_2w - \Sigma yf_2w\Sigma f_1f_3w$ in R10,

$b = (S(R16)-R5R11+R9R14)/R7 = (S\Sigma f_1f_3w-B\Sigma yf_1w+Q\Sigma f_1^2w)/D$ in R3,

$c = (R6R11-R10R15-R8R14)/R7 = (C\Sigma yf_1w-S\Sigma f_1f_2w-R\Sigma f_1^2w)/D$ in R4.

Closeness of Fit

The number of data sets, n, is stored by LBL10 in R7. Then 22 + (2n + 22.02)/1000 is used as a y index in R20. The x index in R21 is larger by 1. R9 and 10 are cleared. LBL14 uses RCL IND21 to retrieve next x value which is used by LBLC to calculate $\hat{y}$. RCL IND20 provides y;

y $\hat{y}$-y is printed after incrementing R20. $|\hat{y}-y|$ is added to R10 and $(\hat{y}-y)^2$ to R9. **R21** is incremented. As long as the integer portion in **R21** is not greater than 22 + 2n, the calculation will return to the beginning of LBL14. After the last set has been treated, ISG**21**will cause a jump to division of R10 by n and division of R9 by n-2 followed by the square root.

Note 3 of the Directions section of Chapter III indicates a procedure for printing of the weighted average $\sum_i w_i|\hat{y}_i-y_i|/\Sigma w_i$. Minor modifications of LBL14 can allow calculation of $\sqrt{\Sigma w_i(\hat{y}_i-y_i)^2/(\Sigma w_i-2)}$ or simultaneous calculation and printing of both weighted and unweighted quantities.

Retrieval of (x_i,y_i) Values

All inserted data sets are stored in R22 to R21+2n and can be retrieved by step (7). n User A calculates the y,x index 22+(19+2n)/1000 for R21 and LBL6 uses LBL16 to retrieve y_1 and x_1 which are then inserted by LBLE. This continues until the next to last set has been processed when ISG21 causes jump to add 0.1 to R21. LBL16 retrieves the last y,x set and LBLE completes insertion of the last data set. The rest of the Procedure is the regular least-square calculation for the indicated equation. As indicated in step (7) of Chapter III, a minor modification allows insertion of weighting factors. Step (7) thus allows repeated least-square treatments of the data using different fitting equations and different weighting factors.

Special Programs

The introductory text of Chapter III discusses equations for fitting enthalpy, heat capacity, and partial molal data. Their application will be illustrated by some examples.

If it is desired to fit drop calorimetry data, $y = (H_T-H_i)/R$, as a function of $\theta = T-T_i$, where T_i is the reference temperature of the calorimeter, with a smooth joining to C_P and dC_P/dT at T_i obtained from low temperature calorimetric measurements, program ab2 can be used with the following functions.

$f(y) = y/\theta - C_{P,i}/R - \frac{1}{2}\theta(dC_P/RdT)_i$, $f^{-1}(y) = \theta[f(y)+C_{P,i}/R] + \frac{1}{2}\theta^2(dC_P/RdT)_i$,

$x'=\theta$, $f_1(x') = \theta^2/T_i^3T$, $f_2(x') = (1/3)\theta^2$. The data for αCa will be used to illustrate the insertion of the functions and testing of the program.
$T_i = 298.15$, $(C_p/R)_{298} = 3.16$, $(d(C_p/R)dT)_{298} = 1.2 \times 10^{-3}\ K^{-1}$, n=5, I=100.

(1b) XEQ '□a□b□2' → F2, EEX 2 STO 00, 'SIZE'032, 298.15 STO 19, 6 EEX 4 CHS STO21

(2) With no entries from previous use of the program,

PRGM	x⇆y	STO6	RCL19	-	STO5	/	3.16	-	RCL5
RCL21	X	-	RCL 6	x⇆y	SST	SST	3.16	+	RCL5
RCL21	X	+	RCL5	X	SST	SST	SST	RCL19	-
SST	□LAST x	3	□y^x	/	RCL5	X	RCL6	/	SST
SST	SST	x^2	3	/	SST	SST	SST	0	X
PRGM									

(3b) 300 ↑ 6.0 User E 400, 326.6 E 500, 651.7 E 600, 979.3 E 700, 1307.9 E 800

(4) R/S c = -16 805, SST d = -6.7×10^{-9}

(5b) 300 C 5.85, R/S 326.4, R/S 651.8, R/S 979.4, R/S 1307.8

To calculate the other constants and C_p/R, the following additions to to program are made to the end of LBL 16.

□GTO16	PRGM	□GTO.543	□RTN	□LBL'B'	RCL2	RCL19	3	□y^x	/
RCL3	RCL19	X	-	2	X	RCL21	2	X	+
STO4	RCL3	RCL19	X	+	RCL19	X	CHS	RCL2	RCL19
□x^2	/ -	3.16	+	STO1	□RTN	RCL4			
LBLD	↑	↑	↑	RCL3	X	RCL4	+	X	RCL1
+	RCL2	R↑	□x^2	/	+	□RTN	R↑	RCL00	+
GTOD	PRGM								

User B a = 3.3688 b = -6.414×10^{-5}
$C_p/R = 3.369 - 6.4 \times 10^{-5}T - 6.7 \times 10^{-9}\ T^2 - 16\ 805 / T^2$.
300 D 3.162, R/S 3.237, R/S 3.268, R/S 3.281, R/S 3.286, 298.15 D 3.160, 350 D 3.208

There is some uncertainty in the value of dC_p/dT at 298.15K. The retrieval capability of step (7) is illustrated by repeating the fit with $(d(C_p/R)/dT)_{298} = 1.202 \times 10^{-3}K^{-1}$ instead of $1.2 \times 10^{-3}K^{-1}$.

(1b) XEQ '□a□b□2'→ F2, EEX 2 STO 00.

As R21 which had been used to store dC_p/RdT is used in step 7, R21 in f(y) and f'(y) must be changed to R1. Also the use of R21 in LBLB must be changed to RCL1. As above, $\frac{1}{2}(d(C_p/R)/dT)_{298}$ is stored.

(2) PRGM □GTO.045 ← RCL1 □GTO.055 ← RCL1 □GTO.557 ← RCL1 PRGM

(7) 6.01 EEX 4 CHS STO1 5 User A 800

(4) R/S c = -16 920, SST d = 4.6×10^{-9}

User B a = 3.3731, SST b = -7.757×10^{-5}

$C_p/R = 3.373 - 7.76 \times 10^{-5}\ T + 4.6 \times 10^{-9}\ T^2 - 16\ 920 / T^2$

300 D 3.162, R/S 3.237, R/S 3.268, R/S 3.281, R/S 3.286

If $(dC_p/dT)_i$ is not well known, but $C_{P,i}$ is known, the drop calorimeter data can be treated by abc2 as described in the introductory text of Chapter III

$$f(y) = y/\theta^2 - C_{P,i}/R\theta,\ f^{-1}(y) = \theta^2 f(y) + \theta C_{P,i}/R$$

$$f_1(x') = \theta,\ f_2(x') = (\theta + T_i)^{-1}.$$

(1c) XEQ '□a□b□c□2', EEX2 STO 00.

(2) With no entries from a previous calculation,

PRGM	x⇄y	STO6	298.15	-	STO5	□x^2	/	3.16	RCL5
/	-	RCL6	x⇄y	SST	SST	RCL5	X	3.16	+
RCL5	X	SST	SST	SST	298.15	-	SST	SST	SST
SST	←	RCL6	1/x	SST	SST	SST	0	X	PRGM

(3b) 300 ↑5.85 User E 400, 326.4 E 500,651.85 E 600, 979.4 E 700, 1307.8 E 800

(4) R/S $a_0 = -2.54 \times 10^{-3}$, SST $a_1 = 3.09 \times 10^{-6}$SST, $a_{-1} = 1.10$

(5) 300 User C 5.85, R/S 327.4, R/S 649.6, R/S 974.8, R/S 1314.6

REFERENCES

(1) L. Brewer, Estimation of Thermodynamic Data and Phase Diagrams Using HP-65 Calculator Programs, LBL-4994, June 1976.

(2) L. Brewer, HP-67 Calculator Programs for Thermodynamic Data and Phase Diagram Calculations, LBL-5485, May 1978.

(3) M. Abramowitz and I.A. Stegun, Editors, Handbook of Mathematical Functions, N.B.S. Applied Mathematics Series 55, June 1964, Supt. of Documents, U.S. Gov't Printing Office, Washington.

(4) R. Hamming, Numerical Methods for Scientists and Engineers, McGraw-Hill, New York, 1973.

(5) W.E. Wentworth, J. Chem. Educ. 42, 96-103, 162-7 (1965).

(6) W.E. Deming, Statistical Adjustment of Data, John Wiley, New York, 1943.

(7) C.H. Shomate, J. Am. Chem. Soc. 66, 928 (1944).

(8) T. Chiang, Y.A. Chang, Can. Metall. Quart. 4, 233-41 (1975).