ACTA CHEMICA SCANDINAVICA 16 (1962) 173–191

High-speed Computers as a Supplement to Graphical Methods

II. Some Computer Programs for Studies of Complex Formation Equilibria

NILS INGRI and LARS GUNNAR SILLÉN

Department of Inorganic Chemistry, Royal Institute of Technology, Stockholm 70, Sweden

A series of programs for high-speed electronic computers have proved useful in investigations on equilibria with polynuclear complexes. The programs are so constructed, that, when a new system is to be studied, only a small part (the "SP") needs to be rewritten (and this is easily done) whereas the main part (the "HP") can be used without change. It could also be used for problems from other fields, chemical or not.

The programs KUSKA, PROKAUS, PROKAIS, HALTA and LETAGROP are given in full — written in Ferranti Mercury Autocode — their principles are discussed, and some initial difficulties of a general interest are pointed out. These programs solve the problems indicated in (6), in the text.

LETAGROP may be called a generalized least-squares method for finding a set of unknown constants from experimental data, even with a non-linear and implicit functional relationship. The principles of LETAGROP were indicated in part I [1].

The studies of complicated solution equilibria in this laboratory have required a fair amount of computational work. At first, the primary measurements (emf's, analyses, volumes) are reduced to the type of data, say (Z, a, B), (b, a, B), most suited for mathematical treatment. From these data, and with no previous assumption as to which out of all conceivable complexes A_pB_q are important, it is usually possible by a mainly graphical treatment to deduce the formulas and approximate formation constants of the main products, or at least to exclude all but a few possibilities. This part of the work is relatively simple.

The hard work, however, comes in with the refinement of the equilibrium constants, and the attempts to deduce formulas and stability constants for species present in smaller amounts. For the construction of the graphs it is often necessary to solve a large number of equations of, say, 6th, 7th or 12th

degree; if only desk computers are available, the best but still very time-consuming way is to use auxiliary graphs.

Since electronic high-speed computers were used to a large extent in the crystallographic work in this laboratory, it was natural to apply such computers to the computations of solution chemistry. Indeed, during the last year or so a Ferranti-Mercury computer has been used by workers of this laboratory to calculate projection maps[1] and species distribution diagrams for polynuclear systems, thus saving, for each system studied, weeks of drudgery and making it possible to increase the accuracy of the result.

The time-determining factor, which had earlier been the calculation of the points on the projection maps, now became the plotting of the points from the tables calculated by the computer. Especially for systems with many species, the refinement of the equilibrium constants required a large number of maps, which sometimes became a little difficult to survey. Since the standard "least squares" treatment (see, *e.g.*, Rydberg [2]) cannot be applied in our systems with polynuclear species, the program LETAGROP [3] was worked out in early 1961, which made possible a simultaneous refinement of many constants.

Notes on the teaching of computer programming

In the meantime, many more of the chemists in this laboratory learned how to program a computer by means of seminars and lectures in the department of inorganic chemistry, by learning from others, and by reading for themselves. The Ferranti-Mercury computer, which was used for most of our calculations of solution equilibria, has a very good and easily understandable automatic code [4], the "autocode". It seems possible in 4 hours lectures + 4 hours problem class to teach students how to write simple programs in the autocode, if one applies the same principles as when teaching a new language. One should rather begin by giving simple sentences and making it clear what they mean than by first studying the muscles of the tongue and larynx, or memorizing the grammar, or memorizing a dictionary.

One must only first realize that the computer contains a number of boxes (registers), each of which has a label like A, B, C_4, *etc.*, and contains a number which is zero until changed by order. Some boxes, the "indexes" I, J...S, T can only contain integers between -512 and $+511$ whereas the others can contain numbers between $\approx 10^{-70}$ and 10^{70}.

Practically the only other thing the chemist needs to know about the inside of the machine is that there is a little demon in it, the "computer", who can do simple arithmetical calculations very rapidly and who can place a new number in a given box, read the content of any box, and send letters and numbers to be printed, if he is told to do so in correct computer language.

At first we did not intend to publish our programs in detail since we thought that anybody who can write down a set of mathematical equations can also make a computer program for them. We have, however, come to think that it might be useful to publish some programs in full, and tell about our experiences since this might save time for other chemists, and help them to overcome initial hesitations and start programming for themselves.

The programs will be written in the Ferranti Mercury autocode [4] but should not be hard to translate to other computer codes.

Main parts of the program

The program that we give the computer to read, as a punched tape, in general consists of three parts, which are usually separate tapes, the first two ending with the instruction "→"; which means: "stop and hoot, and you will get more tape to read".

1. HP (huvudprogram) the main program.
2. SP (särprogram) the special program.
3. The data.

The HP has a name, usually derived from Swedish terms: Kuska, Proka, Halta, Letagrop. Each HP has been so devised that it can be applied to a general class of problems whereas everything that is special in a problem is stated in the SP. Thus, when one passes to a new chemical system, or a new hypothesis, only the SP need be rewritten, which is very easy, and there is no need to think through the functioning of the HP.

The data are the equilibrium constants, concentrations, *etc.*, needed in the calculations on a special problem.

Equations for polynuclear complexes

The programs Kuska, Proka and Halta were primarily designed for calculations on equilibria with polynuclear complexes; with a proper SP, however, they can also be applied to very different problems.

If two reagents, A and B, form a series of complexes A_pB_q, the law of mass action gives for each of these complexes

$$c_{pq} = [A_pB_q] = \beta_{pq}a^pb^q \qquad (1)$$

where a and b are the concentrations of free A and B. The mass balance gives [5] for the total (analytical) concentrations A and B, and the average number of A bound per B, Z:

$$B = b + \Sigma\, qc_{pq},\ A - a = BZ = \Sigma\, pc_{pq} \qquad (2)$$

In dealing with hydrolysis reactions, one often counts ($-H^+$) as the ligand [6]; then, a is replaced by h^{-1} in (1) whereas in (2)

$$BZ = h - H = \Sigma\, pc_{pq} \qquad (2a)$$

In eqns. (2) and (2a), the sums are taken over all sets (p, q).

In translation to computer language, it would be helpful to use symbols that are as similar as possible to those used by equilibrium chemists. However, a few changes must be made since the computer does not distinguish between small and big letters (upper and lower case letters), nor can it use double indices:

Chemical	$A(-H)$	B	Z	BZ	$a(h^{-1})$	b	c_{pq}	β_{pq}	$\log \beta_{pq}$
Computer	A	B	Z	C	U	V	c_i	b_i	e_i

The "chemical" symbols in parentheses are for the special ligand ($-H^+$). Although the usual typescript of the program uses only upper case letters, and sets the index on the main line, we have chosen for readability sometimes to use lower case letters and subscripts, even when we render computer language.

There are no special symbols for p and q; their numerical values for each complex follow from the SP.

With the symbols of the computer (but not yet correct computer language) eqns. (1) and (2) would read:

$$b_i = 10^{e_i} = \exp(e_i \ln 10) \; ; \; c_i = b_i U^p V^q \tag{3}$$

$$B = V + \Sigma q c_i \; ; \; BZ = C = \Sigma p c_i \tag{4}$$

In addition we have

$$A - U = BZ \tag{5}$$

or for the special ligand ($-H^+$)

$$A + 1/U = BZ \tag{5a}$$

The problems solved by the various programs are as follows:

Known	Searched for	Program	Table
B, U	V, *etc.*	Kuska	1
Z, V	B, U, *etc.*	Prokais	2
Z, U	B, V, *etc.*	Prokaus	3
A, B	U, V, *etc.*	Halta	4

(6)

The "known" quantities may be read one by one from the data tape on instructions like: "read B;". Preferably they are calculated as members of an arithmetical series with first term (say, a_1) and increment (say, a_2) given; then one uses a cycle instruction of the type: "$K = 0(1)R \; ; \; Z = a_1 + Ka_2$ ..... repeat;". In Tables 1—4 the programs are written for the most common way of giving the "known" quantities in each case. This part is easily rewritten if it should be necessary.

The SP contains the relationships $B(U, V)$ for Kuska, and in addition $C(U, V)$ for Prokais and Prokaus, and $A(U, V)$ for Halta. There is nothing in the main program that restricts the functions $B(U, V)$, *etc.*, to be of the forms given by (3) and (4) above. In fact, the same HP could equally well be adapted to an SP with any other functions of the two variables U and V.

The example below with Prokais and Prokaus will illustrate difficulties that may occur with functions that do not increase regularly with both U and V, and how these difficulties can be remedied.

In Tables 1—5, " ; " stands for "carriage return, line feed", the conventional division between two instructions.

Some practical details — such as the controlling indices used in various places — will not be discussed in the following but will be easily recognized by the reader who follows the program in detail.

Table 1. KUSKA.

HP

Title ; KUSKA ; chapter 0 ; $a \rightarrow 6$; $b \rightarrow 10$; $c \rightarrow 10$; $d \rightarrow 3$; $e \rightarrow 10$; $y \rightarrow 4$; newline ;
$H = \Psi\log(10)$; jump 1 ;
2) read(N) ; print(N)2,0 ; read(a_1) ; print(a_1)2,2 ; read(a_2) ;
print(a_2)2,2 ; read(R) ; print(R)3,0 ; read(E) ; print(E)1,5 ; newline;
$i = 1(1)N$; read(e_i) ; print(e_i)1,3 ; $b_i = \Psi\exp(He_i)$; repeat ; newline ;
7) read(B) ; print(B)1,5 ; $y_1 = EB$; newline ; $M = 0$;
$K = 0(1)R$; $W = a_1 + Ka_2$; $U = \Psi\exp(HW)$; print(W)2,2 ; $D = 1$; $V = D$; $G = 2$; jump 3 ;
4) $y_2 = \Psi\mathrm{mod}(B_0 - B)$; jump 5, $y_1 > y_2$; jump 8, $B > B_0$;
$G = 0.5$; $D = GD$; $V = V - D$; jump 3 ;
8) $D = GD$; $V = V + D$; jump 3 ;
6) repeat ; newline ; jump 7 ; → ;

SP

1) print(1.1)1,1 ; print(1.3)1,1 ; print(2.4)1,1 ; newline ; jump 2 ;
3) $c_1 = b_1UV$; $c_2 = b_2UVVV$; $c_3 = b_3UUVVVV$; $C_0 = c_1 + c_2 + 2c_3$;
$B_0 = V + c_1 + 3c_2 + 4c_3$; jump 4 ;
5) print(C_0/B)1,3 ; $M = M + 1$; jump 6, $2 > M$; newline ; $M = 0$; jump 6 ;
Ψexp ; Ψlog ; close ;

Data

3	−8.0	0.1	40	0.0001	; 5.27	7.31	13.50
N	a_1	a_2	*R*	*E*	e_1	e_2	e_3
0.6	0.4	0.2	0.1	0.05	0.02		
B	*B*	*B*	*B*	*B*	*B*		

KUSKA

Table 1 gives the HP, and an example of an SP for Kuska. Name from *kurvska*ra — family of curves. The specific example is from studies of the reactions of $B = B(OH)_3$ with $A = OH^-$ in 3M NaBr. In the SP given, the three complexes AB, AB_3, A_2B_4 ($B(OH)_4^-$, $B_3O_3(OH)_4^-$, $B_4O_5(OH)_4^{2-}$) were considered. The expressions for the equilibrium concentrations of the various complexes are readily recognized after label **3**) in the SP, and also the expressions for $BZ(C_0)$ and $B(B_0)$, The log β_{pq} are given as e_1 through e_3. The problem is to calculate, for a given B, Z as a function of $\log U = \log[OH^-]$. B is given a series of values, which are taken from the experiments. For each B, one sets $\log U = a_1 + Ka_2$, where K goes from 0 to R.

N is the number of complexes, E is the desired accuracy, B_0 and C_0 are the values for B and C calculated with a tentative value for V.

Already during the input of the HP and SP, the computer gives printing order for the title KUSKA, assigns boxes for $a_0 \rightarrow a_6$, $b_0 \rightarrow b_{10}$, *etc.*, and stores the remainder of the program in its quick memory. At "close" it stops, and when restarted begins to read instructions at "chapter 0". Then it calculates ln 10 for future use, has the values $p.q$ for the various complexes printed as decimal fractions (label **1** in SP), reads and prints a number of quantities from the data strip (label **2** in HP), and calculates the b_i from the logarithms e_i.

The main principle of the program is given in Fig. 1a, and still more abbreviated in Fig. 1b. After B has been read from the tape, the required accuracy $y_1 = EB$ is calculated, the first value of U is set, and V guessed. Then the machine begins to adjust V by means of the two "loops":

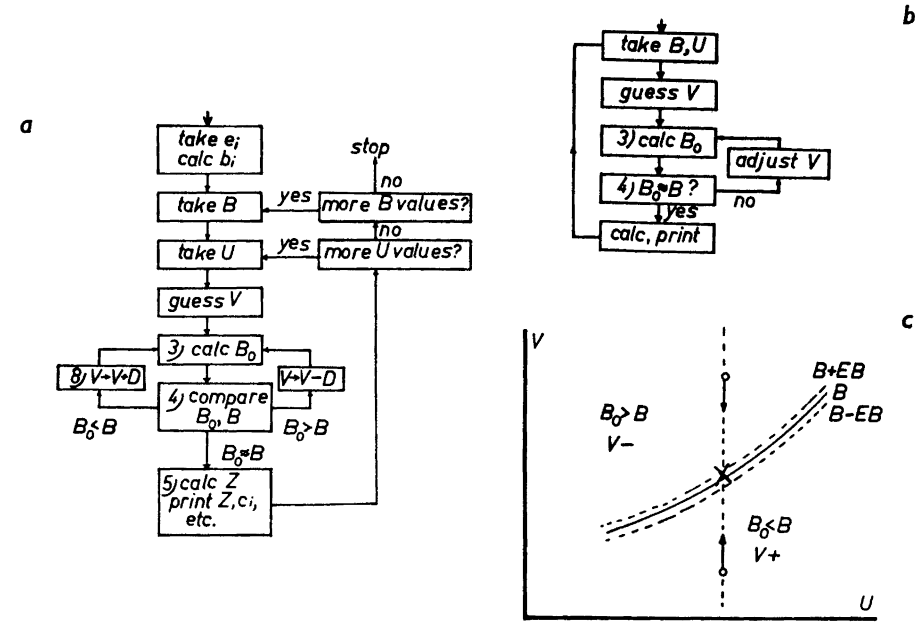

Fig. 1. a) Schematic flow-sheet for central part of program "KUSKA" (Table 1).
b) Same, abbreviated.
c) Adjustment of V for constant U in KUSKA. On the curves $V(U)$ shown, $B_0(U,V) = B$ and $= B \pm EB$.

3) $c_1 = b_1 UV$; $c_2 = b_2 UVVV$; $c_3 = b_3 UUVVVV$; $C_0 = c_1 + c_2 + 2c_3$;
$B_0 = V + c_1 + 3c_2 + 4c_3$; jump 4 ;

4) $y_2 = \Psi \mathrm{mod}(B_0 - B)$; jump 5, $y_1 > y_2$; jump 8, $B > B_0$;
$G = 0.5$; $D = GD$; $V = V - D$; jump 3 ;

8) $D = GD$; $V = V + D$; jump 3 ;

In this part of the program V is systematically varied (adjusted) until $y_2 = |B_0 - B| < EB = y_1$; after that, printing order is given ("jump 5"). D, the correction to V, is doubled (note that at first $G=2$) until V has become too large. After that, D is halved ($G=0.5$) at each turn and is added or subtracted from V depending on whether B_0 is less or larger than B.

To illustrate, if the correct value for V were 59.8, a series of estimated values for V and the following corrections $\pm D$ might be as follows:

1(+2) 3(+4) 7(+8) 15(+16) 31(+32) 63(−16) 47(+8) 55(+4) 59(+2) 61(−1) 60(−0.5) 59.5(+0.25) 59.75(+0.125) *etc.*

When the desired adjustment has been obtained, the values for U, V, all c_i, B and BZ are known so that whatever quantities are needed may be printed. In the special case, only Z was asked for; by using the index M, a more economic use of the paper is achieved. (Six values are printed on the same line).

Table 2. PROKAIS.

```
HP
Title ; PROKAIS ; Chapter 0 ; a→6 ; b→10 ; c→10 ; d→3 ; e→10 ; y→4 ;
   newline ; H=Ψlog(10) ; a5=Ψexp(145) ; a6=Ψexp(−145) ; jump 1 ;
2) read(N) ; print(N)2,0 ; read(a1) ; print(a1)1,3 ; read(a2) ; print(a2)1,3 ; read(R) ;
   print(R)3,0 ; read(a3) ; print(a3)1,2 ; read(a4) ; print(a4)1,2 ; read(S) ; print(S)3,0 ;
   read(E) ; print(E)1,6 ; newline ;
7) i=1(1)N ; read(ei) ; print(ei)1,3 ; bi=Ψexp(Hei) ; repeat ; newline ;
K=0(1)R ; Z=a1+Ka2 ; print(Z)1,3 ; newline ;
M=0 ;
L=0(1)S ; V=Ψexp(LHa4−Ha3) ; D=1 ; U=D ; G=2 ; jump 3 ;
4) jump 6, B0>a5 ; jump 6, a6>B0 ; y1=B0Z−C0 ; y2=Ψmod(y1) ; y3=EB0 ;
   jump 5, y3>y2 ; jump 8,y1>0 ;
G=0.5 ; D=GD ; U=U−D ; jump 3 ;
8) D=GD ; U=U+D ; jump 3 ;
6) repeat ;
newline ; repeat ;
newline ; jump 7 ; → ;

SP
1) print (1.1)1,1 ; print(1.3)1,1 ; print(2.4)1,1 ; newline ; jump 2 ;
3) c1=b1UV ; c2=b2UVVV ; c3=b3UUVVVV ; C0=c1+c2+2c3 ;
   B0=V+c1+3c2+4c3 ; jump 4 ;
5) W=Ψlog(U) ; print(W/H)3,3 ; W=Ψlog(B0) ; print (W/H)1,3 ; M=M+1 ;
   jump 6, 3>M ; newline ; M=0 ; jump 6 ;
   Ψexp ; Ψlog ; close ;

Data
   3    0.1  0.1  6    2    0.2  16   0.0001 ;  0    0    0.9 ;
   N    a1   a2   R    a3   a4   S      E       e1   e2   e3

   0    0    0.6 ; 0    0    0.3 ; 0    0    0 ; 0    0   −0.3  0    0   −1 ;
   e1   e2   e3    e1   e2   e3    e1   e2   e3   e1   e2   e3   e1   e2   e3

   0    0   −1.6 ;
```

After printing, another value of U, and eventually another value of B, is taken.

The procedure is illustrated in Fig. 1c, which gives curves $V(U)_{B_0}$ thus curves with B_0 constant in the coordinate plane (U, V). The adjustment is satisfactory and the calculation ended when B_0 comes within the limits $B \pm EB$.

PROKA

In the names Prokais and Prokaus, "Proka" comes from the Swedish *pro*jektions*kar*ta = projection map and "S" from säkring (safety catch).

Tables 2 and 3 give the HP for Prokais and Prokaus. Table 2 also gives an example of an SP (which is the same for Prokais and Prokaus) and of a data strip.

The SP shown refers to the same system, $OH^- + B(OH)_3$, and the same complexes as in Kuska (**3** and **1**).

The Proka programs can be used for calculating a series of projection maps [1] $\log B(\log U)_Z$ for comparison with the experimental data, and estimation of equilibrium constants. Since a normalized projection map is required, two of the equilibrium constants can arbitrarily be set equal to 1 (thus the

Table 3. PROKAUS.

HP

Title ; PROKAUS ; Chapter 0 ; $a \rightarrow 6$; $b \rightarrow 10$; $c \rightarrow 10$; $d \rightarrow 3$; $e \rightarrow 10$; $y \rightarrow 4$;
newline ; $H = \Psi\log(10)$; $a_5 = \Psi\exp(145)$; $a_6 = \Psi\exp(-145)$; jump 1 ;
2) read(N) ; print(N)2,0 ; read(a_1) ; print(a_1)1,3 ; read(a_2) ;print(a_2)1,3 ; read(R) ;
print (R)3,0 ; read(a_3) ; print(a_3)1,2 ; read(a_4) ; print(a_4)1,2 ; read(S) ;
print(S)3,0 ; read(E) ; print(E)1,5 ; newline ;
7) $i = 1(1)N$; read(e_i) ; print(e_i)1,3 ; $b_i = \Psi\exp(He_i)$; repeat ; newline ;
$K = 0(1)R$; $Z = a_1 + Ka_2$; print(Z)1,3 ; newline ;
$M = 0$;
$L = 0(1)S$; $U = \Psi\exp(LHa_4 - Ha_3)$; $D = 1$; $V = D$; $G = 2$; jump 3 ;
4) jump 6, $B_0 > a_5$; jump 6, $a_6 > B_0$;
$y_1 = B_0Z - C_0$; $y_2 = \Psi\mathrm{mod}(y_1)$; $y_3 = EB_0$; jump 5, $y_3 > y_2$; jump 8, $0 > y_1$;
$G = 0.5$; $D = GD$; $V = V - D$; jump 3 ;
8) $D = GD$; $V = V + D$; jump 3 ;
6) repeat ;
newline ; repeat ;
newline ; jump 7 ; → ;

SP

1) print(1.1)1,1 ; print(1.3)1,1 ; print(2.4)1,1 ; newline ; jump 2 ;
3) $c_1 = b_1UV$; $c_2 = b_2UVVV$; $c_3 = b_3UUVVVV$; $C_0 = c_1 + c_2 + 2c_3$;
$B_0 = V + c_1 + 3c_2 + 4c_3$; jump 4 ;
5) $W = \Psi\log(U)$; print(W/H)3,3 ; W $= \Psi\log(B_0)$; print (W/H)1,3 ; $M = M + 1$;
jump 6, $3 > M$; newline ; $M = 0$; jump 6 ;
Ψexp ; Ψlog ; close ;

Data

3		0.55	0.05	9		0.00	0.20	15		0.0001	;	0		0		0.9	;						
N		a_1		a_2	R		a_3		a_4	S		E				e_1		e_2		e_3			
0		0		0.6	; 0		0		0.3	; 0		0	0	;	0		0		−0.3	; 0		0		−1 ;
e_1		e_2		e_3	e_1		e_2		e_3	e_1		e_2	e_4		e_1		e_2		e_3	e_1		e_2		e_3
0		0		−1.6	;																		

corresponding $e_i = \log \beta_{pq} = 0$). To Z a series of values $Z = a_1 + Ka_2$ are given, where K goes from 0 to R. For each value of Z, log V (in Prokais) or log U (in Prokaus) passes through a series of values $(La_4 - a_3)$ where L goes from 0 to S.

The general principle of Prokais is shown in Fig. 2a, (flowsheet) and Fig. 2b. For each set (Z, V), U is adjusted until the ratio C_0/B_0 is close enough to the Z desired. If $C_0/B_0 < Z$, then U is increased, otherwise decreased. This procedure converges as rapidly as Kuska in case $Z(U)_V$ increases over the whole range, as shown in Fig. 2b.

In some systems, however, $V(U)_Z$ has a maximum (Fig. 2c). This behavior was first met with in the borate system, for $Z > 0.5$. If V is chosen larger than the maximum value, C_0/B_0 is always $< Z$, and the computer goes on increasing U until some number (one of the c_i, B_0 or C_0) becomes larger than the largest allowable number in the computer ($2^{256} \approx 10^{77}$) and the machine stops.

In order to avoid machine stops for this reason, there is a safety exit after label **4**: if B_0 would become larger than $a_5 = e^{145} \approx 10^{63}$ or smaller than $a_6 = e^{-145}$, the machine stops trying to solve the equations and jumps to the next set (V, Z).

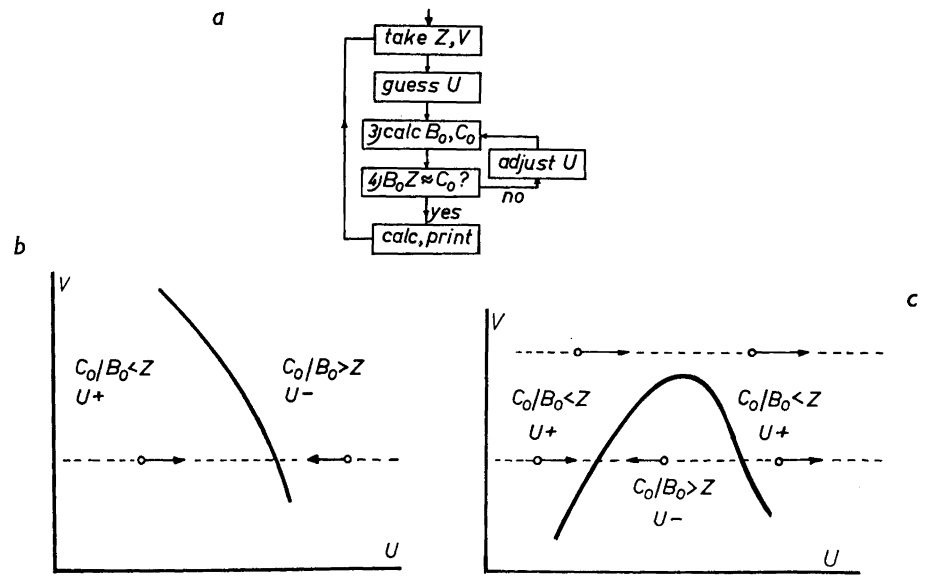

Fig. 2. a) Schematic flow-sheet for central part of "PROKAIS" (Table 2).
b) Adjustment of U for constant V with PROKAIS, to approach desired value Z for C_0/B_0 (thick curve).
c) Behavior of PROKAIS for curve with a maximum.

Even for values smaller than the maximum V, at a given Z Prokais can only find U values that correspond to the left branch of the curve, in Fig. 2c. If one happens to pass beyond the right-hand branch, U will increase till the safety catch is triggered.

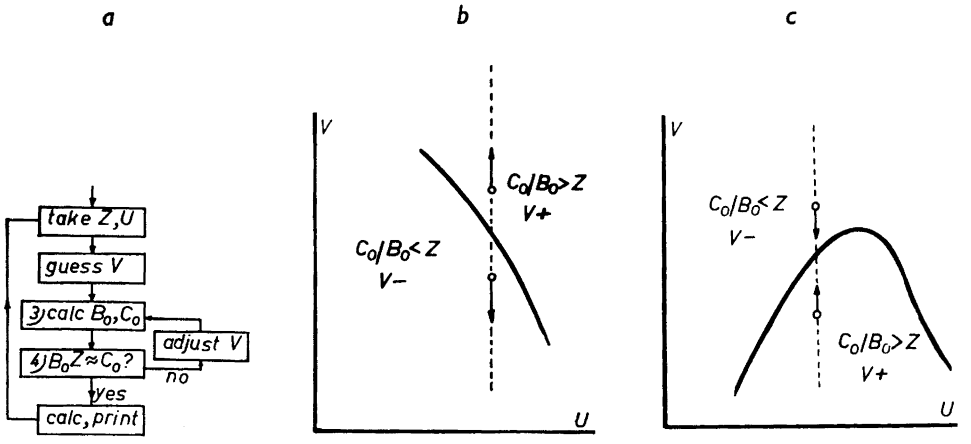

Fig. 3. a) Schematic flow-sheet for central part of "PROKAUS" (Table 3.).
b) Behavior of PROKAUS for curve shown in Fig. 2 b (no adjustment possible).
c) Adjustment of V for constant U with PROKAUS for curve shown in Fig. 2 c.

For systems of this type, the program Prokaus (Table 3, Figs. 3a—c) has been devised: for various sets (Z, U) it adjusts V, increasing V if C_0/B_0 is too high and decreasing V if C_0/B_0 is too low. This program will give the whole of a curve like that in Fig. 3c, but no part of a curve like that in Fig. 3b. For a system where part of the ranges behaves like Figs. 2c—3c, one should apply both Prokais and Prokaus, and so obtains supplementary parts of the projection map, overlapping to some extent.

Table 4. HALTA.

HP

```
Title ; HALTAPIFF ; Chapter 0 ; a→6 ; b→10 ; c→10 ; d→3 ; e→10 ; f→3 ; g→4 ; u→4 ;
   v→4 ; w→4 ; y→4 ; newline ; H=Ψlog(10) ; jump 1 ;
2) read(N) ; print(N)2,0 ; read(E) ; print(E)1,8 ; read(G1) ; print(G1)3,0 ; read(T) ;
   print(T)2,0 ; newline ;
i=1(1)N ; read(ei) ; print(ei)1,3 ; bi=Ψexp(Hei) ; repeat ; newline ;
7) read(B) ; print(B)1,5 ; newline ; y0=EB ; y=y0G1 ;
read(a1) ; print(a1)1,5 ; read(a2) ; print(a2)1,5 ; read(R) ; print(R)2,0 ; newline ;
U=0.1B ; V=0.1B ;
K=0(1)R ; A=a1+a2K ; print(A)1,5 ; Q=0 ;
14) P=0 ; U0=U ; D0=2 ; D2=V ; G=2 ; J=0 ; jump 3 ;
4) y1=B−B0 ; y2=Ψmod(y1) ; y3=A−A0 ; y4=Ψmod(y3) ;
   J=J+1 ; jump 17, J>100 ; jump 10, 1>D0 ; jump 9, y0>y2 ;
15) jump 8, y1>0 ; G=0.5 ; D2=GD2 ; V=V−D2 ; jump 3 ;
8) D2=GD2 ; V=V+D2 ; jump 3 ;
9) jump 5, y>y4 ; P=P+1 ; VP=V ; Q=Q+1 ; D0=0 ; D1=U ; G=2 ; J=0 ;
10) jump 11, y0>y4 ; jump 12, y3>0 ;
G=0.5 ; D1=GD1 ; U=U−D1 ; jump 3 ;
12) D1=GD1 ; U=U+D1 ; jump 3 ;
11) jump 5, y>y2 ; P=P+1 ; UP=U ; Q=Q+1 ;
jump 5, Q>T ; jump 13, P>3 ;
D0=2 ; D2=V ; G=2 ; J=0 ; jump 15 ;
13) F0=2U2−U0−U4 ; U3=U4−U2 ; V2=V3−V1 ; W3=U3/F0 ; U=U4+U3W3 ;
   V=V3+V2W3 ; Q=Q+1 ;
jump 16, 0>U ; jump 16, 0>V ; jump 14 ;
16) print(16)2,0 ; newline ; U=U4 ; V=V3 ; jump 14 ;
17) print(17)2,0 ; newline ; jump 11, 1>D0 ; jump 9 ;
6) repeat ;
newline ; jump 7 ;→ ;
```

SP

```
1) Title ; URAN 14 APRIL 1961 ; print(2.2)1,1 ; print(4.3)1,1 ;
   print(5.3)1,1 ; print(6.4)1,1 ; newline ; jump 2
3) W=UUV ; c1=b1VW ; c2=b2c1W ; c3=b3c1UW ; c4=b4c1c1UU ;
   A0=−1/U+2c1+4c2+5c3+6c4 ; B0=V+2c1+3c3+4c4 ; jump 4
5) W=A0+1/U ; Z=W/B0 ; print(1000Z)3,3 ; W=Ψlog(U) ;
   print(W/H)1,5 ; W=2000c1/B0 ; print(W)3,3 ; W=4000c2/B0 ;
   print(W)3,3 ; W=5000c3/B0 ; print(W)3,3 ; W=6000c4/B0 ;
   print(W)3,3 ; print (Q) 3,0 ; newline ; jump 6
   Ψexp ; Ψlog ; close ;
```

Data

```
4      0.000001   100   33   −6.03   −7.17   −10.52   −7.36  ;  0.100   0.050
N        E        G1    T     e1      e2      e3       e4        B       a1

−0.004   30  ;  0.075  0.040  −0.003  30  ;  0.050  0.030  −0.002  30  ;  0.025
 a2      R        B      a1     a2      R        B      a1     a2      R       B
```

HALTA

Table 4 gives the HP for a member (Haltapiff) of the program family Halta, (*halt-ta*beller = concentration tables) which is intended to calculate U and V, and thus all concentrations c_i, in cases where only A and B are known. The SP and the data are for the hydrolysis of $B = UO_2^{2+}$; $A = -H^+$. (Calorimetric titrations of Dr. Kurt Schlyter).

This program has, among other applications proved useful for calculations of the concentrations of various species in calorimetric titrations with polynuclear equilibria.

The SP and the data refer to the hydrolysis of $B = UO_2^{2+}$, $A = -H^+$. It is assumed that (in the range studied) only the complexes A_2B_2, A_4B_3, A_5B_3, and A_6B_4 need to be considered; (compare list at SP label **1**). In the present case, the variable $W = U^2V$ has been introduced for simplicity. The equilibrium concentrations $c_1 ... c_4$ of the four complexes are easily recognized at label **3** of the SP.

When U and V, and thus the c_i are known, the printing order (SP, label **5**) in this case starts with 1 000 Z, with three decimals. The following numbers are: log U(= $-\log [H^+]$), and $2c_1/B$, $4c_2/B$, *etc.* = the number of A per B bound in the form of A_2B_2, A_4B_3, *etc.*

B has a series of values, which are here given by the experiments. For each B, one uses a series of values, $A = a_1 + Ka_2$, where K goes from 0 to R. The first, and more subordinate, part of the HP can be rewritten to meet special needs, for instance, so that A and B are varied simultaneously, as happens in some titrations.

The flow-sheet of the first edition of Halta is given in Fig. 4a. The procedure is depicted in Fig. 4b, which gives, in the (U, V) plane, the two curves on which the functions A_0 and B_0 have the desired values, $A_0 = A$ and B_0

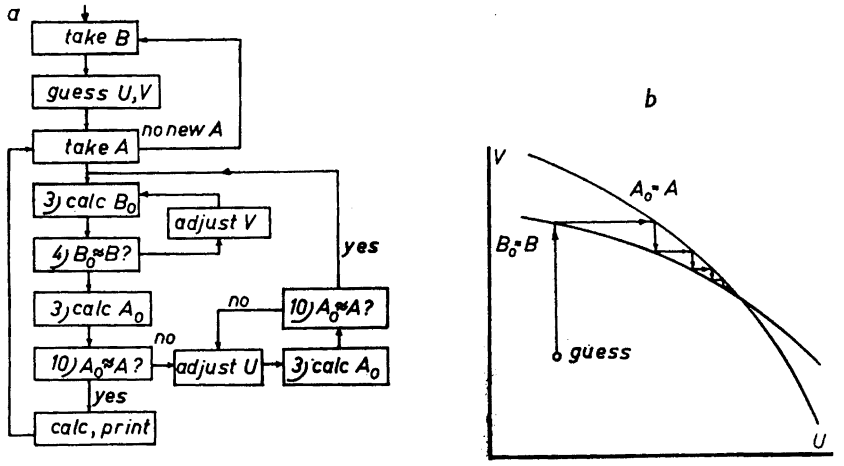

Fig. 4. a) Schematic flow-sheet of original "HALTA" (complete program not given here)
b) Performance of original HALTA in approaching the desired point (intersection point between $B_0(U, V) = B$ and $A_0(U, V) = A$).

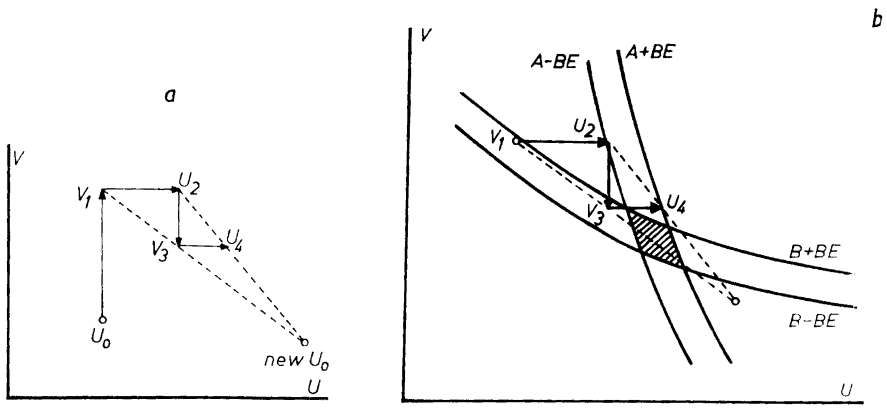

Fig. 5. a) "Shot" for a new point in "HALTAPIFF" (Table 4).
b) "Miss" because of bad aim. The shaded area is the "bull's eye", where both equations $B_0 = B$ and $A_0 = A$ are fulfilled within the desired accuracy.

$= B$. First, U is kept constant and V adjusted to make $B_0 = B$, then V is kept constant and U adjusted to make $A_0 = A$, *etc.* In this way, one approaches the intersection point of the two curves. The convergence was sometimes rather rapid (a matter of a few seconds, thus something like 3—5 turns between the curves). However, in some cases the machine could keep working for several minutes before it found the intersection point; it always did find it, but at a high cost of computer time. The performance was followed in detail, and in one case, where the angle between the two curves was small, 210 attempts were counted before the intersection point was found within the accuracy prescribed; in other cases the number may have been still higher.

In order to save time we let the machine "shoot" its way, (Fig. 5a). After it had found four points, two on each curve, it calculated the point of intersection of two straight lines through them (label **13**, Table 4), and used this point as its next guess. This speeded up the approach considerably, but the computer invented a number of practical jokes. Sometimes the machine stopped because the shot had hit a point with negative U or V; this was easily avoided by adding a new "safety catch" (label **16**).

In a number of cases, the computer was stuck in an infinite loop so that it kept working on a single point for minutes without result (Fig. 5b). The reason was found to be that for each adjustment one does not reach the curve $B_0 = B$ or $A_0 = A$ exactly but is certain only to come within the limits $B_0 = B \pm BE$ and $A_0 = A \pm BE$, thus in a band surrounding the correct curves (in the program, $y_0 = BE$). The closer one is to the intersection point, the larger will be the risk that the small deviations will give a bad aim, so that the "shot" may even take one further away from the intersection point than one was originally. Therefore, the computer was aiming and shooting madly to hit the bull's-eye but the hit-points bounced away.

As a remedy, the requirements on fit were sharpened by decreasing E; on the other hand, the "bull's-eye" was increased by giving printing order at a larger tolerance, $y = G_1BE$, each time one had solved the equation for B_0 and first tried to see if the one for A_0 was satisfied (labels **4→9→5**). The calculations were shortened in most cases, but eventually the computer could again get stuck in a loop. This might probably have been remedied by decreasing E and increasing G_1 further, but instead the calculation was interrupted after a certain number, T, of attempts; this is controlled by a counting index Q. The number of attempts Q, is printed as a check (label **5**).

LETAGROP

Part 1 of this series of papers [3] described a general principle for adjusting a number of unknown constants k_r from measurements of the function y, containing the constants searched for and a number of known quantities a_1, a_2... The principle is that the (weighted) error-square sum is calculated for a number of points, as a function of the unknown constants (which are here considered as variables). The error-square sum is approximated as a second-degree equation and its minimum searched for. By using the "D boundary", one can also derive expressions for the "standard deviation" of the various constants.

Table 5. LETAGROP.

HP
Chapter I ; Title ; LETAGROP ; $a\rightarrow 13$; $b\rightarrow 13$; $c\rightarrow 13$; $d\rightarrow 13$; $e\rightarrow 13$; $F\rightarrow 120$;
$g\rightarrow 13$; $h\rightarrow 13$; $U\rightarrow 120$; $v\rightarrow 13$; $X\rightarrow 2$; $Z\rightarrow 120$;
1) $a'\approx 1$; $x'\approx 101$; $y'\approx 201$; $v'\approx 1001$; $w'\approx 2001$; $b'\approx 3001$; $c'\approx 4001$; $d'\approx 5001$;
jump 7, $L>0$; jump 2, $M=1$; jump 3, $M=2$; $X_0=X$; $j=1(1)R$; read(g_j) ;
read(h_j) ; repeat ; $j = 1$;
5) $M=1$; $s=\Psi$intpt(g_j) ; $e_s=e_s+h_j$; across 3/0 ;
2) $X_1=X$; $e_s=e_s-2h_j$; $M=2$; across 3/0 ;
3) $X_2=X$; $e_s=e_s+h_j$; jump 4, $X_2>X_1$; $h_j=-h_j$; $w=X_1$; $X_1=X_2$; $X_2=w$;
4) $a_j=0.25X_2-0.25X_1$; $d_j=0.25X_1+0.25X_2-0.5X_0$; $j=j+1$; jump 6, $j>R$; jump 5 ;
6) $\Psi_7(1)a_1$, R ; $j=0$;
11) $j=j+1$; jump 12, $j>R$;
$L=1(1)j$; $v_L=0$; repeat ; $v_j=d_j$; jump 10, $j=R$; $s=\Psi$intpt(g_j) ; $e_s=e_s+h_j$;
$L=j+1$;
8) $t=\Psi$intpt(g_L) ; $e_t=e_t+h_L$; across 3/0 ;
7) $v_L=0.5X-0.5X_0+a_j+a_L-d_j-d_L$; $e_t=e_t-h_L$; $L=L+1$; jump 9, $L>R$; jump 8 ;
9) $e_s=e_s-h_j$;
10) $O=R_j-R$; $\Psi_7(1001+O)v_1$, R ; jump 11 ;
12) $a=R$; $w'=\Psi_{16}(1001,a,a)$; $c\approx aa$; $b'=\Psi_{11}(1001,2001,c)$; $c'=\Psi_{17}(4001,a)$;
$c'=\Psi_{28}(3001,R,R)$; $x'=\Psi_{26}(4001,1,a,1,a)$; $y'=\Psi_{26}(101,1,1,1,a)$; $\Psi_6(201)y,1$
$\Psi_6(101)v_1$, R ; $X=X_0-y$; jump 14, $X>0$; $X=-X$; print(-1)1,1 ; newline ;
14) $w=Q-P+1-N$; $y=X/w$; $w=\Psi$sqrt(y) ; print(w)1,4 ; newline ;
$j=1(1)R$; $s=\Psi$intpt(g_j) ; print(s)2,0 ; $e_s=e_s+h_jv_j$; print(e_s)2,3 ;
$O=Rj-R+j$; $\Psi_6(4000+O)w,1$; $X=wy$; jump 13, $0>X$; $w=\Psi$sqrt(X) ; $X=wh$;
print(X) 1,3 ;
13) newline ; repeat ;
newline ; $'c'=\Psi_{15}(5001,c)$; across 14/0 ;
Ψsqrt ; close ;
Chapter 0 ; variables 1 ; $H=\Psi$log(10) ; jump 1 ;
2) read(Q) ; $K=1(1)Q$; read(F_K) ; read(U_K) ; read(Z_K) ; repeat ; *stop* ; jump 14, $R=-2$;
15) read(N) ; $i=1(1)N$; read(e_i) ; repeat ; read(E) ;

14) read(R) ; jump 15, $R=-1$; jump 2, $-1>R$; read(P) ; read(Q) ; $j=0$; $L=0$; $M=0$; $K=P$; jump 3 ;
6) $V=D$; $G=2$; $E_0=EB$; jump 7 ;
8) $D_0=\Psi\mathrm{mod}(B_0-B)$; jump 10, $E_0>D_0$; jump 9, $B>B_0$; $G=0.5$; $D=GD$; $V=V-D$; jump 7 ;
9) $D=GD$; $V=V+D$; jump 7 ;
11) $w=Z_0-Z_K$; $X=X+G_0ww$; jump 12, $R>0$; newline ; print(K) 2,0 ; print(F_K)1,5 print(U_K) 1,3 ; print(Z_K) 1,3 ; print(Z_0-Z_K) 1,4 ;
12) $K=K+1$; jump 13, $K>Q$; jump 5 ;
13) print(10000X) 3,4 ; print($Q+1-P$) 3,0 ; $K=P$; newline ; jump 14 ; $R=0$; across 1/1 ; → ;

SP

1) Title ; borat 3MNaClO$_4$; print(1.1)1,1 ; print(1.3)1,1 ; print(1.5)1,1 ; newline ; jump 2 ;
3) $X=0$; $i=1(1)N$; $b_i=\Psi\exp(He_i)$; print(e_i)1,3 ; jump 4, $i\neq 6$; newline ;
4) repeat ;
5) $B=F_K$; $w=14.22-U_K$; $U=\Psi\exp(Hw)$; $D=0.001$; jump 6 ;
7) $c_1=b_1UV$; $c_2=b_2UVVV$; $c_3=b_3UVVVVV$; $B_0=V+c_1+3c_2+5c_3$; jump 8 ;
10) $C_0=c_1+c_2+c_3$; $Z_0=C_0/B_0$; $G_0=1$; jump 11 ;
Ψexp ; close ; → ;

MD

90	;	0.01	;	6.283	;	0.08	;	0.01	;	6.342	;	0.07	;	;
Q		F_1		U_1		Z_1		F_2		U_2		Z_2		...

DS

3	;	−9.005	;	−6.911	;	−6.620	;	0.0001	;
N		e_1		e_2		e_3		E	

3	;	1	;	90	;	1	;	0.02	;	2	;	0.05	;	3	;	0.05	;
R		P		Q		g_1		h_1		g_2		h_2		g_3		h_3	

0	;	1	;	90	;
R		P		Q	

Table 5 gives an HP, which is applicable under rather general conditions, an SP, devised for a specific case from borate equilibria, and an example of data, which consist of two parts: the experimental data MD (mätdata) and DS ("dagens spaning, = day's orders for searching").

It has been necessary to change the symbols a little from part I, partly to use the language of the computer, partly to use symbols similar to those used in the older programs Kuska, Proka and Halta.

The experimental data are assumed to be given as triplets of numbers (F_K, U_K, Z_K), with K from 1 to Q. The present program provides for at most 120 such triplets thus $Q \leq 120$. (This is ample for most work on mononuclear complexes, but requires some random screening for most of our systems with polynuclear complexes).

The unknown quantities, N in number, are denoted by $e_1 ... e_N$, and one is assumed to know the function

$$Z_0 (F_K, U_K, e_1 ... e_N) \approx Z_K \tag{7}$$

The only error present is assumed to be the random error in Z_K; if a systematic error is suspected, it may be treated as another unknown constant e_i.

To each point measured one can assign a weight G_0, which can be expressed as a function of known quantities; this expression, like everything else that is special for the problem, is given in the SP.

The symbols in part I have been translated as follows to computer language:

Part I	k_r	N	y_i	f	a_{1i}	a_{2i}	n	w_i	U
Computer	e_i	N	Z_k	Z_0	U_K	F_K	Q or $(Q-\mathrm{P}+1)$	G_0	X

Calculating the error square sum

After stopping at "close", the computer begins to read the instructions at "Chapter 0": it calculates $H = \ln 10$, prints the various $p.q$ as before (**1**/0)*, reads the Q triplets of data (label **2**/0) and a set of approximate values for $e_1 \ldots e_N$ — calculated by graphical methods, estimated or guessed — and then it searches for the error square sum. At first (**3**/0 SP), the e_i may be recalculated to quantities (here the b_i) more suitable for the calculations. The following treatment differs depending on whether (7) is given explicitly or not.

Implicit equations. The program is specially designed for cases with two equations:

$$B = B_0(U,V,e_1 \ldots e_N)\,;\; Z_K = Z_0(U,V,e_1 \ldots e_N) \qquad (8,9)$$

Here, B and U are functions of F_K and U_K which are given in the SP, label **5**/0. Note that (9) differs from (7) in that V is unknown and must be eliminated using (8).

In the present case, Z_K is simply the number of OH bound per $B(OH)_3$, $B = F_K$ is the total boron concentration, $U_K = \log [OH^-] = \log U$, so that $U = 10^{U_K}$. The equations (SP, **7**/0) are easily recognized as expressing the law of mass action, and the mass balance.

First, V is calculated from the set (B, U) using a Kuska loop (**8**, **9**/0). Then, Z and the weight G_0 are calculated (SP, **10**/0), the squared error multiplied by the weight, $G_0(Z_0 - Z_K)^2$, is added to the error square sum X(HP, **11**/0), and a new value for K is taken. When all points have been worked through (**13**/0), 10 000 times the error square sum, 10 000 X, is calculated and printed, together with the number of points $(Q + 1 - P)$.

Explicit equations. If, on the other hand, (7) is given explicitly, the expression for Z can be calculated straightforwardly in the SP between (**5**/0) and "jump 11", without any need to go through a Kuska loop. One should, however, set out the labels **7**) and **10**) at some harmless point, if one does not care to delete the Kuska in the HP. (The computer does not accept "jump" instructions, even if void, without a corresponding label).

If the data consist of sets of two values (as for instance in the adjustment of a straight line), $U_K = 0$ may be written everywhere and not used, or the HP may be rewritten for data pairs.

*1/0" means label 1 in chapter 0.

Table 6. Example of results with LETAGROP.

LETAGROP

1.1	1.3	1.5		
−9.005	−6.911	−6.620	4.0083	90
−8.985	−6.911	−6.620	4.8430	90
−9.025	−6.911	−6.620	4.8535	90
−9.005	−6.861	−6.620	6.8642	90
−9.005	−6.961	−6.620	7.8930	90
−9.005	−6.911	−6.570	4.1038	90
−9.005	−6.911	−6.670	4.1076	90
−8.985	−6.861	−6.620	8.7998	90
−8.985	−6.911	−6.570	4.9771	90
−9.005	−6.861	−6.570	7.7148	90

0.0021

1	−9.007	0.005
2	−6.903	0.008
3	−6.648	0.047

−9.007	−6.903	−6.648				
1	0.01000	6.283	0.080	−0.0010		
2	0.01000	6.342	0.070	−0.0004		
3	0.01000	6.434	0.054	0.0027		
4	0.01000	6.521	0.046	0.0018		
5	0.01000	6.659	0.033	0.0015		
6	0.01000	6.808	0.024	0.0007		
—	—	—	—	—		
—	—	—	—	—		
90	0.60000	7.568	0.095	−0.0039	3.9416	90

The first table gives e_1, e_2, e_3, X and Q. Then comes the standard deviation of Z, 0.0021; the "best" values of e_1, e_2 and e_3 with their standard deviations; and a table of the data (B, log U, Z) ending with the deviation in Z.

At last comes the error square sum with these "best" values, and the number of points, Q.

Variation of the e_i

All the operations described hitherto have been made in Chapter 0 of HP + SP. When the error-square sum, X, has been calculated (**13**/0), the computer passes across to **1**/1, thus label **1** in Chapter 1, and begins to vary the constants systematically; for each new set of e_i it goes back to chapter 0 to calculate the error square sum X.

Through DS the computer has been told how many (R) constants it should vary (**14**/0), and after (**1**/1) it is told their indices ($g_1 ... g_R$), and the steps to vary them in ($h_1 ... h_R$).

It may be helpful to compare Table 5 with Table 6, which gives an example of the results, in a special case. The first line in Table 6 gives the first approximate values of the constants (let us call them e_i') and the X value (X_0) obtained with them.

The first constant e_s ($s = g_1$ is here = 1) is first set at $e'_s + h_1$ (**5**/1), the error sum calculated and called X_1 (**2**/1), e_1 set at $e_s' - h_1$, the error sum calculated and called X_2, and e_s reset at e_s' (**3**/1). If the second error-square sum,

X_2 is smaller than the first, e_s' should probably be decreased; then h_1 is replaced by $(-h_1)$, and X_1 and X_2 change places. This is done in order to have the mixed points (see below) as close as possible to the minimum point. — This is repeated for all the e_i to be varied.

For each e_j (**4**/1) the quantities a_j and d_j are calculated, which are related to the constants in the general equation (part I, eqn. 42)

$$U = c_0 + 2\Sigma c_{0r}x_r + \Sigma\Sigma c_{rs}x_r x_s \qquad \text{(I:42)}$$

With our notations we have

$$x_j = e_s - e_s'; \ (s = g_j) \qquad (10)$$

$$X_0 = c_0; \ X_1 = c_0 + 2\,c_{0j}h_j + c_{jj}h_j^2;$$

$$X_2 = c_0 - 2c_{0j}h_j + c_{jj}\,h_j^2 \qquad (11)$$

Comparing eqn. (11) and Table 5 (**4**/1) we see that

$$a_j = 0.25\,X_2 - 0.25\,X_1 = -\,c_{0j}h_j \qquad (12)$$

$$d_j = 0.25\,X_1 + 0.25\,X_2 - 0.5\,X_0 = 0.5\,c_{jj}h_j^2 \qquad (13)$$

When this procedure has been repeated with all the constants e_i to be varied, the R terms a_j are accumulated in the auxiliary store from box No. 1 on (**6**/1).

Now (**11**/1,**7**/1) from box 1001 (v') on, a $R \times R$ matrix is laid out, row by row, which will contain the elements ("v_L")

$$\begin{aligned} v_{Lj} &= 0 \text{ for } L<j \\ v_{Lj} &= d_j = 0.5\,c_{jj}h_j^2 \text{ for } L = j \\ v_{Lj} &= c_{Lj}h_L h_j \text{ for } L<j \end{aligned} \qquad (14)$$

The "mixed" terms are obtained by calculating X for $e_s = e_s' + h_j$, $(x_j = h_j, \ s = g_j)$ and $e_t = e_t' + h_L$, $(x_L = h_L, \ t = g_L)$.

This error-square sum, from (I:42) becomes

$$X = c_0 + 2c_{0j}h_j + 2c_{0L}h_L + c_{jj}h_j^2 + c_{LL}h_L^2 + 2c_{Lj}h_L h_j =$$

$$X_0 - 2a_j - 2a_L + 2d_j + 2d_L + 2c_{Lj}h_L h_j$$

$$c_{Lj}h_L h_j = 0.5\,X - 0.5\,X_0 + a_j + a_L - d_j - d_L \qquad (15)$$

Thanks to the instructions (label **3**/1), the e_i sets with two constants varied will usually be on the same side of e_i' as the minimum of the pit.

Matrix operations

From (**12**/1) on begins a series of matrix operations. The mirror image of the matrix (14) is calculated and stored from box 2001 (w') on. The sum of the two matrices, which means the $R \times R$ quadratic form, $c_{Lj}h_L h_j$, is calculated and stored from box 3001 (b') on. The reciprocal of the latter is calculated and stored from box 4001 (c') on by instruction Ψ_{28}.

Multiplication with the $1 \times R$ matrix a_j gives (eqn. I:43a) the R corrections x_j'' to the e_s' to be varied, in units of h_j. They are stored from box 101

(x') on. In box 201 (y') is stored the sum $-\Sigma c_{0_j} x_j''$, and then X at the minimum point (eqn. I:44), thus U_0, is calculated. For each constant to be varied, the corrected value is calculated by addition of $h_j v_j$, and the standard deviation is calculated and printed (eqn. I:41).

The last operation, $c' = \Psi_{15}$ (5001, c) uses the zeros in the empty boxes from 5001 on to "sweep" c' for the next use.

Some of the more surprising "≈" and "Ψ intpt" instructions in Table 5 have proved necessary since the computer is rather particular on having indexes in some contexts and "variables" in others. Reference is made to the manual [4] for details on matrix operations.

Check. Since there may be many pitfalls in working out a program like this, we have checked the program on some textbook examples of finding the parameters of a straight line. LETAGROP at once found the same answers (parameters and standard deviation) as the textbook, either it started with a bad or a good guess. This is expected, since with a linear equation, the error square sum is exactly a secondary degree function.

Procedure

A treatment with the program Letagrop gives, starting from an approximate set of constants, in general an improved set, and estimates of their standard deviations. If the standard deviation of the data, σ comes out much higher than expected, usually something is wrong with the input of the data, or with the selection of the function (7).

If the function is linear, the minimum in the error squares' sum is found at once, however bad the first guess. On the other hand, for nonlinear functions, the first approximation found may be in error and successive attempts must be made. It is our experience that two or three approximations will in general suffice, if the first guess was reasonably good. The further one is from the minimum the more important become higher-degree terms. The matrix term C_{rr}/C (eqn I:46) that occurs in the standard deviation D_r^2 may sometimes come out as negative. To prevent the machine from stopping, a safety exit has been inserted, so that only an empty space is left, as an indication that the surface misbehaves. This may happen for very bad guesses, or for erroneous functions.

It will be seen from Table 5 how the index R is used for giving new orders. If R is a positive number, it means the number of constants to be varied in a new attempt. For $R = 0$, the computer uses the last "best" set of constants to calculate and print the deviation $Z_0 - Z_K$ for each of the experimental points. As a check of the input of the data, the experimental data (F_K, U_K, Z_K) are again printed (**11**/0).

For $R = -2$ new data are offered instead of the earlier ones (**14—2**/0), whereas the SP and the e_i are retained. With $R = -1$, (**14—15**/0) the data are kept but the constants are changed. With $R = -3$, both data and constants are changed.

By using the instruction "rmp" = "read more program", one may make the computer read a new "chapter 0", thus let it start on a new problem with-

out the necessity for reading chapter 1 again. This may save some time in studying series of problems with different SP.

A program as given in Table 5 is limited to 120 triplets of data, and 13 unknown to be varied. It would not be hard to rewrite the program, using the auxiliary memory, or rearranging the quick memory, in order to increase the number of unknown constants, or data sets.

For some reason, the matrix operations do not seem to work with $R = 1$. So, with only one constant to be varied one would have to add a special program, which we shall do on the day we meet with so simple a problem.

Examples of the results will be given in following papers.

Acknowledgements. We wish to thank Dr. Germund Dahlqvist and Mr. Per Erik Persson, fil.kand., for teaching us the elements of computer programming, and for helping us when we were in trouble. The technical staff of the Ferranti-Mercury computer of AB Atomenergi have always been very helpful. We would specially like to thank Mr. Assar Åkesson. Our friends in the Department of inorganic chemistry have helped us in many ways and provided the right mixture of patience and enthusiasm in the periods when the programs were worked out.

This is part of a program supported by *Statens Naturvetenskapliga Forskningsråd (Swedish National Science Research Council)*, *Statens Råd för Atomforskning (Swedish Atomic Energy Research Council)* and by the *Office of Scientific Research* of the Office of Aerospace Research, USAF, through its European Office on Contract No AF 61–(052)162.

REFERENCES

1. Sillén, L. G. *Acta Chem. Scand.* **10** (1956) 803.
2. Sullivan, J. C., Rydberg, J. and Miller, W. F. *Acta Chem. Scand.* **13** (1959) 2023.
3. Sillén, L. G. *Acta Chem. Scand.* **16** (1962) 159 (Part I).
4. Brooker, R. A., Richards, B., Berg, E. and Kerr, R. H. *The Manchester Mercury Autocode system*, University, Manchester 1959.
5. Sillén, L. G. *Acta Chem. Scand.* **8** (1954) 299.
6. Hietanen, S. and Sillén, L. G. *Acta Chem. Scand.* **8** (1954) 1607.

Received June 21, 1961.