ARKIV FÖR KEMI Band 31 nr 26

4.68 12 16 Read 9 October 1968

High-speed computers as a supplement to graphical methods. 7

Model selection and rejection with LETAGROP. Elimination of species with negative or "insignificant" equilibrium constants

By Lars Gunnar Sillén and Björn Warnqvist

ABSTRACT

The program LETAGROP may be used to find the set of adjustable parameters $\vec{\mathbf{k}}$ that minimizes a certain error square sum $U(\vec{\mathbf{k}})$. Some of these parameters (for instance equilibrium constants) may, by definition, have only positive or zero values, and LETAGROP may then be told to protect them from becoming negative. If some protected parameters have negative values at the minimum of the calculated second-degree surface $U(\vec{\mathbf{k}})$, deduced from a "shot", a new version of the procedure MIKO searches systematically for the set $\vec{\mathbf{k}}_0'$ that gives the lowest value for U and still fulfills the condition that each protected k_i is either $=0$ or >0. Alternatively, the condition may be set that $k_i=0$ or $>F\sigma\sigma(k_i)$, where $F\sigma$ is a chosen "rejection factor", and $\sigma(k_i)$ is the standard deviation. This alternative is used in a "species selector" STYRE. To an initial set of species STYRE may add one after another from a list of species to be tested. If a new species improves the error square sum U for the given data and fulfills the $F\sigma$ condition, then it is accepted otherwise rejected. Species from the initial set may be rejected in the process. The test for the "final" set of species is that no new species are accepted when all the rejected ones are recycled through STYRE.

The general minimizing program LETAGROP (parts 1–4, 6) is designed to find the set of parameters $\vec{\mathbf{k}}_0$ that minimizes a function $U(\vec{\mathbf{k}})$ of these parameters, typically an error square sum. Some of these parameters may, by definition, be positive or zero but never negative; an important case is equilibrium constants. Such parameters $k[ik]$ will be called "protected", and in the computer they are marked by setting a corresponding Boolean *posk*[*ik*]: =**true**. (For non-protected parameters, *posk*[*ik*] is **false**.)

In the course of a "shot", only sets of non-negative k_i values are used in the actual calculation of the necessary number of points on the surface $U(\vec{\mathbf{k}})$; this is regulated by procedure PLUSKA (part 6). However, when the position of the minimum of the second-degree surface through these points is calculated in GROP, some protected parameters may turn negative. As discussed in part 3, it is not enough to change the value of the negative parameters to zero, thus to make a projection of the calculated minimum; one must really find the minimum for $U(\vec{\mathbf{k}})$ in the "reduced space", which term we shall use for the (generalized) plane in $(U, \vec{\mathbf{k}})$ space where the "negative" parameters are set equal to zero. The fundamental necessary equations (3–5 below) were given in part 3; we shall apply them here in a slightly different way.

The method outlined in part 3, and used in the block MIKO in part 4, worked well in the majority of cases, namely when only one parameter turned negative. Sometimes, however, in the first calculation of the minimum several protected parameters turned negative, or in reduced space some new parameter turned negative that had been positive in the first calculated minimum. On closer inspection of such cases it was sometimes found that too many parameters had been eliminated; indeed by "saving" some eliminated parameters it was possible to find a still lower value for U.

Hence, the block MIKO was rewritten so that, if more than one of the parameters k_i turns negative in GROP, in the following calculation a systematic attempt is made to save parameters. In other words, MIKO tries to find the point on the calculated second-degree surface $U(\vec{\mathbf{k}})$ that has the lowest value for U while all parameters k_i are positive or zero.

The same set of calculations proved useful also for "species selection" ("model selection", part 3, p. 1095–1097). When one tries to explain a certain set of data by some set of species, the "best" values of the formation constants β (called k below) for some species, even if positive, may come out quite low as compared with its standard deviation $\sigma(k)$. One may then ask whether the value for k is "significant" or not. Depending on one's judgment and experience, one may define for each problem a "rejection factor" F_σ such that a k value is counted as "significant" only if

$$k > F_\sigma \sigma(k) \tag{1}$$

Species which do not fulfill (1) are hence rejected. This means that, given the data and a tentative list of species, one sets the problem to find the set of k values that gives the lowest value for U while still for each species

$$k > F_\sigma \sigma(k) \text{ or } k = 0 \tag{2}$$

We shall see now the block MIKO can be used for solving this type of problem, too.

Equations for MIKO

We shall here, for simplicity, consider only cases where all parameters are protected, and mainly consider the N parameters that are really adjusted.

Let us recall that after the calculations in LETA and GROP, one may express the calculated second-degree surface $U(\vec{\mathbf{k}})$ in the form:

$$U - U_0 = (\vec{\mathbf{k}} - \vec{\mathbf{b}})\,\mathbf{A}(\overset{\downarrow}{\mathbf{k}} - \overset{\downarrow}{\mathbf{b}}) \tag{3=3:35}$$

where

$$\mathbf{A} = ((\mathbf{SH})^{\mathrm{T}})^{-1}\mathbf{R}(\mathbf{SH})^{-1} \tag{4=3:35a}$$

$\vec{\mathbf{b}} = \vec{\mathbf{k}}_0$ is the calculated minimum and $\mathbf{R}$ is the matrix of the second-degree terms in $U(\vec{\mathbf{v}})$ (3:7, 3:11, 6:2).

We shall now consider a number of "reduced spaces", that is sections of total space in which some of the parameters k_i are set equal to 0. We shall express each vector $\vec{\mathbf{x}}$ in total space as the sum of two vectors $\vec{\mathbf{x}}_+$ and $\vec{\mathbf{x}}_-$: in $\vec{\mathbf{x}}_+$ the components of $\vec{\mathbf{x}}$ keep their original values along the "plus" coordinates, which means those that remain in the reduced space, whereas the components of $\vec{\mathbf{x}}$ along the eliminated "minus" coordinates are set equal to zero; the other vector $\vec{\mathbf{x}}_-$ contains the "minus" components of $\vec{\mathbf{x}}$ whereas its "plus" components are set equal to zero. Similarly, a two-dimensional matrix is given as the sum of four matrices, $\mathbf{A}_{++}$, $\mathbf{A}_{+-}$, $\mathbf{A}_{-+}$ and $\mathbf{A}_{--}$ (part 3, p. 1093).

The calculated minimum in total space is

$$\vec{\mathbf{k}}_0 = \vec{\mathbf{b}} = \vec{\mathbf{b}}_+ + \vec{\mathbf{b}}_- \qquad (5 = 3{:}36\,\text{b})$$

We have now decided to consider only vectors $\vec{\mathbf{k}} = \vec{\mathbf{k}}_+$ (3:36a) so that only the "plus" components differ from zero. As radius vector in reduced ("plus") space it is convenient to use the distance from the projection of the calculated minimum, hence

$$\vec{\mathbf{d}}_+ = \vec{\mathbf{k}}_+ - \vec{\mathbf{b}}_+ \qquad (6)$$

For $\vec{\mathbf{k}}_+$ in reduced space we find from (6) and (5)

$$\vec{\mathbf{k}}_+ = \vec{\mathbf{b}} + \vec{\mathbf{d}}_+ - \vec{\mathbf{b}}_- \qquad (7)$$

The corresponding U value on the calculated second-degree surface must follow equation (3) which is valid for total space or any section of it. Inserting (7) gives (since $\mathbf{A}$ is symmetrical)

$$U - U_0 = (\vec{\mathbf{d}}_+ - \vec{\mathbf{b}}_-)\,\mathbf{A}(\overset{\downarrow}{\mathbf{d}}_+ - \overset{\downarrow}{\mathbf{b}}_-) = \vec{\mathbf{b}}_-\,\mathbf{A}_{--}\,\overset{\downarrow}{\mathbf{b}}_- - 2\vec{\mathbf{b}}_-\,\mathbf{A}_{-+}\,\overset{\downarrow}{\mathbf{d}}_+ + \vec{\mathbf{d}}_+\,\mathbf{A}_{++}\,\overset{\downarrow}{\mathbf{d}}_+ \qquad (8)$$

On the other hand, (8) should also represent a paraboloid in reduced space, characterized by the coordinates of the minimum

$$\vec{\mathbf{d}}_{0+} = \vec{\mathbf{k}}_{0+} - \vec{\mathbf{b}}_+ \qquad (9)$$

by a symmetrical matrix $\mathbf{A}'$ at first unknown, and by the height of the minimum, U_0'.

The general equation of this paraboloid is

$$U - U_0' = (\vec{\mathbf{d}}_+ - \vec{\mathbf{d}}_{0+})\,\mathbf{A}'(\overset{\downarrow}{\mathbf{d}}_+ - \overset{\downarrow}{\mathbf{d}}_{0+}) = \vec{\mathbf{d}}_{0+}\,\mathbf{A}'_{++}\,\overset{\downarrow}{\mathbf{d}}_{0+} - 2\vec{\mathbf{d}}_{0+}\,\mathbf{A}'_{++}\,\overset{\downarrow}{\mathbf{d}}_+ + \vec{\mathbf{d}}_+\,\mathbf{A}'_{++}\,\overset{\downarrow}{\mathbf{d}}_+ \qquad (10)$$

Since equations (8) and (10) describe the same surface, comparison term by term gives

$$\mathbf{A}'_{++} = \mathbf{A}_{++} \qquad (11)$$

$$\vec{\mathbf{d}}_{0+} = \vec{\mathbf{b}}_-\,\mathbf{A}_{-+}\,(\mathbf{A}_{++})^{-1} \qquad (12)$$

$$U_0' = U_0 + \vec{\mathbf{b}}_-\,\mathbf{A}_{--}\,\overset{\downarrow}{\mathbf{b}}_- - \vec{\mathbf{d}}_{0+}\,\mathbf{A}_{++}\,\overset{\downarrow}{\mathbf{d}}_{0+} \qquad (13)$$

Hence, since $\mathbf{A}$, $\mathbf{b} = \mathbf{k}_0$ and U_0 are known from LETA and GROP, we may use (12), (9) and (13) to calculate the position $\mathbf{k}_0'$ and the height U_0' of the reduced minimum.

Notation

Two important reduced spaces are met with during the action of MIKO:

(1) In the *space of maximum reduction*, all parameters are eliminated that have turned negative during any part of the calculation after GROP.

(2) The space with the *lowest reduced minimum* is the one that gives the lowest minimum value for U (on the calculated surface) where still no protected parameter is negative.

In a certain reduced space, the number of "minus" parameters will be called *Nimi* and the number of "plus" parameters *Niplus*, hence $Nimi + Niplus = N$. For instance, let us consider a case with six parameters ($N_k = 6$) out of which four ($ik = 2$, 3, 4 and 6) are varied ($N = 4$, $ivar[1] = 2$ etc). Let us suppose that in a certain reduced space the parameters corresponding to $ivar[2]$ and $ivar[4]$ are eliminated:

ik	1	2	3	4	5	6	
i		1	2	3		4	(14)
		+	−	+		−	

In this case we would have $Nimi = 2$, $iplus[1] = 1$, $iplus[2] = 3$, $imi[1] = 2$, and $imi[2] = 4$.

In the space of maximum reduction, *Nimi* is called *Nimimax*, and the numbers of the *imi* series are called *imix*[*i*]. The Boolean *minx*[*j*] is set **true** if *j* belongs to the *imix* series, thus if the corresponding parameter is eliminated in the space of maximum reduction. In the space of the lowest reduced minimum, *Nimi* is called *Nimirem*, and the numbers of the *imi* series are called *imirem*[*i*].

The symbols in the equations above are represented in the ALGOL program as follows:

$\mathbf{A}$ = rucka[1:N, 1:N], $(\mathbf{A}_{++})^{-1}$ = are[1:Niplus, 1:Niplus], $\vec{\mathbf{b}}$ = kbom[1:N], $\vec{\mathbf{b}}_{-}\mathbf{A}_{-+}$ = pina[1:Niplus], $\vec{\mathbf{d}}_{0+}$ = dkbom[1:N], $\vec{\mathbf{k}}'_0$ = kbred[1:Niplus], U_0 = Unospar, U'_0 = Uno, $\sigma(k)$ = dard[1:Niplus].

In the space with the lowest reduced minimum, $\mathbf{k}'_0$, U'_0 and $\sigma(k)$ are denoted by krem[1:(N − Nimirem)], Unorem and darm[1:(N − Nimirem)].

Mink is a Boolean which is **true** during the systematic "saving" of eliminated parameters, otherwise **false**. For *ival*, see below.

Short survey of MIKO

A recent version of MIKO is given in Table 1; like STYRE (see below) and the special blocks UBBE in following parts, it is to be inserted as a block in the general ALGOL program given in part 6.

MIKO is entered when some "posk-protected" parameters have turned negative in GROP.

Normally this happens only for common parameters k (thus when *Tage* is **false**) and at present MIKO is applied to group parameters, k_s, only in one special case, met in spectrophotometry: the k_s in this case are the molar absorptivities which cannot be negative. On the other hand, since they appear only in linear equations, $U(\mathbf{k}_s)$ is a true second-degree surface, and it can do no harm if negative values are used in the calculation of U in UBBE. However, a value for U with some $k_s < 0$ will not be recorded by LETA as a minimum $U_{\min}$, since it would not be accepted in the final test.

When MIKO is entered, *rucka* (**A**) is calculated. At Miko1, the space of maximum reduction (*Nimimax*, *imix*, *minx*) is searched for. Then a systematic attempt is made to eliminate only one, only two ... of the *Nimimax* suspected parameters.

Table 1. Block MIKO, to be inserted into LETAGROP main program, part 6.

```
MIKO:      begin real Unorem, Unospar ; integer Nimi, Nimimax, Nimirem, Niplus ;
           real array are[1:16, 1:16], dard,darm,dkbom,kbred,krem[1:16], pina[1:20], rucka, rut3,
                      SHinv [1:16,1:16] ;
           integer array imi, imirem, imix, iplus, ival[1:16] ; Boolean array minx[1:16] ;
procedure  Gropprov ; begin integer Nimi0 ; Bra: = true ; Nimi0: = Nimi ;
           if not (Tage and Skrikut < 0) and Niplus = 0 then output(61, '/2B'ALLES TOT'') ;
           for i: = 1 step 1 until Niplus do begin j: = iplus[i] ; ik: = ivar[j] ;
             if (Tage and kbred[i] < 0) or (posk[ik] and not Tage and (kbred[i] < 0 or (Kassa and
                      dard[i] > 0 and kbred[i] < sigfak × dard[i]))
               then begin Bra: = false ; if not minx[j] then begin minx[j]: = true ; Mink: = false
                      end ; Nimi: = Nimi + 1 ; imi[Nimi]: = j end end ;
           if not Bra then begin if Tage and Skrikut < 0 then goto Tig ;
             if Nimi0 > 0 then begin output (61, ' 2B 'UT'') ;
               for i: = 1 step 1 until Nimi0 do output (61,'2ZD',ivar[imi[i]]) end ;
             output (61,'/2B,'MIKO'') ;
             for i: = Nimi0 + 1 step 1 until Nimi do output (61,'2ZD',ivar[imi[i]]) ;
Tig:         if Mink then Nimi: = Nimi0 end end Gropprov ;
procedure  Inmix ; begin i: = 0 ;
           for j: = 1 step 1 until N do
             if minx[j] then begin i: = i + 1 ; imix[i]: = imi[i]: = j end ;
           Nimi: = Nimimax: = i end Inmix ;
procedure  Inplus ; begin Boolean array plus[1:12] ;
           for i: = 1 step 1 until N do plus[i]: = true ;
           for i: = 1 step 1 until Nimi do plus[imi[i]]: = false ;
           Niplus: = 0 ;
           for i: = 1 step 1 until N do if plus[i] then begin Niplus: = Niplus + 1 ; iplus[Niplus]: = i
                      end end Inplus ;
procedure  Redgrop ; begin if not (Tage and Skrikut < 0) then output(61, '/') ;
           for i: = 1 step 1 until Niplus do for j: = 1 step 1 until Niplus do are[i,j]: = rucka[iplus[i],
                      iplus[j]] ;
           if Niplus > 0 then INVERT (Niplus, are, 1_{10} - 30, det, SING) ;
           for j: = 1 step 1 until Niplus do begin pina[j]: = 0 ;
             for i: = 1 step 1 until Nimi do begin pina[j]: = pina[j] + kbom[imi[i]] × rucka [imi[i],
                      iplus[j]] end end ;
           for j: = 1 step 1 until Niplus do begin w: = 0 ;
             for i: = 1 step 1 until Niplus do w: = w + pina[i] × are[i,j] ; kbred[j]: = kbom [iplus
                      [j]] + w ; dkbom[iplus[j]]: = w end ;
           Uno: = Unospar ;
           for i: = 1 step 1 until Nimi do for j: = 1 step 1 until Nimi do Uno: = Uno + kbom[imi[i]]
                      × kbom[imi[j]] × rucka[imi[i],imi[j]] ;
           for i: = 1 step 1 until Niplus do for j: = 1 step 1 until Niplus do Uno: = Uno - dkbom
                      [iplus[i]] × dkbom[iplus[j]] × rucka[iplus[i],iplus[j]] ;
           SIGGE ;
           for i: = 1 step 1 until Niplus do begin j: = iplus[i] ; ik: = ivar[j] ; w: = sig2y × are[i,i] ;
             if w > 0 then darr2: = sqrt(w) else darr2: = - abs(stek[j]) ;
             dard[i]: = darr2 end end Redgrop ;
procedure  Rem ; begin Unorem: = Uno ; Nimirem: = Nimi ;
           for i: = 1 step 1 until Nimi do begin j: = imirem[i]: = imi[i] ; ik: = ivar[j] ; krem[j]: = 0 ;
                      darm[j]: = if Tage then darks2[Rs,ik] else dark2[ik] end ;
           for i: = 1 step 1 until Niplus do begin j: = iplus[i] ; krem[j]: = kbred[i] ; darm[j]: = dard
                      [i]end end Rem ;
procedure  Skriv; begin if Tage and Skrikut < 0 then goto Tig ;
           output (61, '/ B 'UNO = ',D.6D_{10} + 3D',Uno) ;
           for i: = 1 step 1 until Niplus do begin ik: = ivar[iplus[i]] ; output (61, '/,2B,'K',2ZD,
                      ' = ', - D.3D_{10} + 3D,2B,  'DARR2 = ', - D.3D_{10} + 3D,2B,'AK = '',ik,kbred[i].
                      dard[i]) ;
             if not Tage then AKUT end ;
Tig:       end Skriv ;
```

```
           Nimi: = 0 ; Niplus: = N ; Mink: = false ;
           for i: = 1 step 1 until N do begin ik: = ivar[i] ; minx[i]: = false ;
             if Tage then kbred[i]: = ks[Rs,ik]: = kbom[i] else kbred[i]: = k[ik]: = kbom[i] ;
             iplus[i]: = i; dard[i]: = if Tage then darks2[Rs,ik] else dark2[ik] end ;
           Gropprov ;
           if Bra then begin if Kassa and not Tage and Slusk = 0 then begin
                 for i: = 1 step 1 until N do if dark2[ivar[i]] < 0 then Bra: = false ;
                 Styr: = if Bra then 8 else 6 ; if Bra then Kassdarr: = true ;
                 Kassa: = false end ;
             goto PROVIN end ;
           Unorem: = Unospar: = Uno ;
           for i: = 1 step 1 until N do for j: = 1 step 1 until N do SHinv[i,j]: = SH[i,j] ;
           INVERT(N,SHinv,1$_{10}$ - 30,det,SING)  ;  MULLE(SHinv,ruta,N,N,N,rut3, - 1)  ;
           MULLE(rut3,SHinv,N,N,N,rucka,1) ;
Mikol:     Inmix ; Inplus ; Redgrop ; Gropprov ; Skriv ; if not Bra then goto Mikol ;
           Mink: = true ; Rem ; Nimi: = 0 ;
Uppmi:     Nimi: = Nimi + 1 ; if Nimi = Nimimax then goto Mikout ;
           for i: = 1 step 1 until Nimi do ival[i]: = i ; ival[Nimi + 1]: = 0 ; goto Mikony ;
Runt:      i: = 0 ;
Hoppmi:    i: = i + 1 ; if ival[i] = Nimimax then goto Uppmi ;
           if ival[i + 1] = ival[i] + 1 then goto Hoppmi ;
           ival[i]: = ival[i] + 1 ; for j: = 1 step 1 until (i - 1) do ival[j]: = j ;
Mikony:    for i: = 1 step 1 unti Nimi do imi[i]: = imix[ival[i]] ;
           Inplus ; Redgrop ; Gropprov ; if not Mink then goto Mikol ;
           if Bra then begin Skriv ; if Uno < Unorem then Rem end ;
           goto Runt ;
Mikout:    for i: = 1 step 1 until N do begin ik: = ivar[i] ;
             if Tage then begin ks[Rs,ik]: = krem[i]; darks2[Rs,ik]: = darm[i] end
             else begin k[ik]: = krem[i] ; dark2[ik]: = darm[i] end end ;
           Mink: = false ;
           if Slusk > 2 then begin N: = Nido ; for i: = 1 step 1 until N do ivar[i]: = ivarge[ido[i]] ;
             goto PROVIN end ;
           if Tage or not Kassa then goto PROVIN ;
           Nimi: = Nimirem ; for i: = 1 step 1 until Nimirem do imi[i]: = imirem[i] ; Inplus;
           Bra: = true ; Kassdarr: = false ;
           w: = Nimi × sigfak↑2 × sig2y/Uno  ;  if  Unorem > Umin + w × Umin  then  Bra: = false  ;
           for i: = 1 step 1 until N do begin
             if krem[i] > 0 and darm[i] < 0 then Bra: = false ;
             if krem[i] ⩽ 0  and  darm[i] > 0  then  Kassdarr: = true  end  ;
           Kassa: = false ;
           if not Bra then begin Styr: = 6 ; Kassdarr: = false ; goto PROVIN end ;
           Styr: = 7 ; Ri: = 0 ; Nimi: = Nimirem - 1 ; Niplus: = Niplus + 1 ;
Uppri:     Ri: = Ri + 1 ; j: = 0 ;
           for i: = 1 step 1 until Nimirem do begin j: = j + 1 ;
             if i = Ri then j: = j - 1 else imi[j]: = imirem[i] end ;
           output (61, '// ' DARR AV KASSERADE k[ik] I GROP") ;
           Rj: = iplus[Niplus]: = imirem[Ri] ; Redgrop ; Skriv ; ik: = ivar[Rj] ; k[ik]: = kbred
                   [Niplus] ; dark2[ik]: = dard[Niplus] ; kass[ik]: = true ;
           output (61, '/,22B 'KASS', - D.5D$_{10}$ + 3D,2B,'DARR = ', - D.3D$_{10}$ + 3D,2B 'AK = " ,
                   k[ik], dark2[ik]) ; AKUT ;
           if Ri < Nimirem then goto Uppri ;
           goto STYRE end MIKO ;
```

If some "plus" parameters turn negative when a certain reduced space is tried, procedure Gropprov prints "UT (*ik* numbers of eliminated parameters) MIKO (*ik* numbers of negative parameters)", e.g. "UT 3 4 MIKO 6". (UT = out).

The latter output means that if k_3 and k_4 are eliminated, k_6 comes out negative or too small according to (2). If every adjustable parameter has turned negative at some place in GROP or MIKO, "ALLES TOT" is printed before proceeding.

If the calculated minimum is acceptable (no protected parameters turn negative), then U_0' (*Uno*) and the calculated parameter values are printed. The lowest *Uno* met with (*Unorem*) is remembered and used at the end of the process.

The systematic elimination is done using a series of integers, *ival*, setting *imi*[*i*] = *imix*[*ival*[*i*]], in a way that is reminiscent of the systematic elimination of solid phases in HALTAFALL (part 5). The working of the algorithm at Uppmi-Hoppmi is illustrated by an example with *Nimimax* = 4; the numbers given are the successive values for *ival*[1] through *ival*[*Nimi* + 1]:

(*Nimi* = 1) 10, 20, 30, 40, (*Nimi* = 2) 120, 130, 230, 140, 240, 340, (*Nimi* = 3) 1230, 1240, 1340, 2340.

The case with *Nimi* = 4, 12340, corresponds to the space of maximum reduction for which *Uno* and the parameters have already been calculated. Finally the lowest calculated minimum, *krem*, is tested at PROVA. Only $\sigma(k)$ = *darr2*, or *dard*, is calculated in the reduced pits.

The individual procedures work as follows. Gropprov (pit test) tests if the *posk* conditions or (2) are fulfilled in the calculated pit. If new parameters turn negative, their *minx* are set **true**. Inmix, using the *minx*, calculates *imix* and *Nimimax* for the space of maximum reduction. Inplus calculates *Niplus* and the *iplus* series, using *Nimi* and *imi*. Redgrop calculates the position of a reduced minimum using eqns (11), (12), (9) and (13). Rem records the lowest acceptable *Uno* value hitherto met with. Skriv writes the values for the *k* and their *darr2* in an acceptable reduced minimum.

It should be noted that all these operations are carried out only on the calculated surface, given by equation (3). Just as in other calculations with LETAGROP there is the possibility that terms of higher than second degree in the real function $U(\vec{\mathbf{k}})$ make the calculated minimum deviate significantly from the real one, expecially if the calculations are carried out on points far from the minimum.

At any rate, the calculated lowest acceptable minimum is tested in PROVA, accepted if it is better than anything met with, or corrected by SLUSS, if a better minimum had been met with earlier. In general, the result will be a lower value for U, and if several shots are ordered, the parameters eliminated in one shot will get another chance in the next one.

Model (species) selection by STYRE

Input for STYRE

A cycle of the model selector is activated by input with the following information: 17(*Rurik*), 1(*Styr*), *sigfak*, *Nskytt*, *Nvar*, $(ik)_{Nvar}$, *Nin*, $(k, (ak)_{Nak}, dark)_{Nin}$ (Input 1).

Styr is a control number. *Sigfak* is the rejection factor F_σ in equation (2) and N_{skytt} is the number of shots to be made for each new combination (new model) before the F_σ criterion (1, 2) is applied. Out of the N_k original common parameters, N_{var} are to be varied = adjusted, namely those with the *ik* numbers given. Often $N_{\text{var}} = N_k$ so that all are adjusted. N_{in} is the number of new common parameters that are to be tested, typically formation constants for new species. For each new parameter k, a guess for its value (usually very crude), the characteristic numbers *ak* (usually "*pot*" = approximate power of ten, and the coefficients p, q, sometimes r, t, *fas*, that describe the composition of the corresponding species) and the first step to be tried (stored as a negative *dark*) are given.

Warning: The sum $N_k + N_{\text{in}}$ must not exceed the maximum number of $k[ik]$ allowed in the declarations: in the program as given in part 6, this is 20, but it could easily be increased.

Another cycle of the species selector may be ordered by the input 17(*Rurik*), 5(*Styr*), *sigfak*, *Nskytt* (Input 5).

Then all the parameters rejected in the preceding cycle get a new chance with, if one wishes, another value for F_σ and another number of shots.

Short survey of STYRE

The action of STYRE (Table 2) may be described by following the changes in the control numbers *Styr*:

With *Styr* = 1, the necessary information (Input 1) is read. Suitable steps for the first N_{var} parameters have either been given after *Rurik* = 3, or 19, or 20, or even calculated by one or more shots before *Rurik* = 17 is set.

As *Styr* passes from (1 or 7) to 3, a new parameter is added to the earlier ones, and first adjusted alone by ENSAM. This gives a reasonable order of magnitude for this parameter and its step (*dark*).

Styr then passes from 3 to 4; the newcomer and the earlier adjustable parameters are adjusted together by *Nskytt* (for instance 1, 2, or 3) shots. During the last of these shots, the program sets the Boolean *Kassa:* = **true**, and uses MIKO to eliminate parameters which did not fulfill the condition (2). If the minimum had an acceptable shape (all *darr* given as positive), MIKO changes *Styr* from 4 to 7 (8). For each rejected parameter it sets the Boolean *kass*[*ik*]: = **true** and remembers its *k* value. Its standard deviation is calculated using a space containing the parameters in the space of the lowest reduced minimum, and in addition that rejected parameter. During this calculation, the Boolean *Kassdarr* = **true**.

On the other hand, if any of the *darr* is given as negative at the lowest reduced minimum, this means that the calculated surface had a saddle point instead of a minimum. MIKO then sets *Styr*: = 6 (from 4). After PROVA the program returns to STYRE where a new shot is set up with half the earlier *stekfak* and *Styr*: = 4, and the calculation is repeated.

The procedure Stuva is used at *Styr* = 1 or 7 to reorganize the parameters in the order:

Ntot (15)

Nk			
Notvar	*N*	*Ntot* − *Nk*	
not *var*[*ik*] not varied	*var*[*ik*], **not** *kass*[*ik*] accepted, varied	**not** *kass*[*ik*] not yet tested	*kass*[*ik*] tested and rejected

The addition of a new parameter may lead to three different results:

(*a*) A new minimum is found that is lower than the earlier one, while all parameter values fulfill (2). Hence the new species is accepted, and none rejected.

(*b*) The new species either gets a negative value for k($U_{\min}$ does not change) or has a positive value, too small to fulfill (2), in which case the improvement of U is approximately:

$$-\delta U_0 \approx \sigma^2(y)\cdot(k/\sigma(k))^2 \approx U_0(k/\sigma(k))^2/(N_{\text{points}}-N) < U_0 F_\sigma^2/(N_{\text{points}}-N) \quad (16)$$

Table 2. Block STYRE, to be inserted into general LETAGROP program, part 6.

```
STYRE:      begin integer Nin, Nvar ;
            switch Styre: = Styr1, Styr2, Styr3, Styr4, Styr5, Styr6, Styr7, Styr8 ;
procedure   Byta (i1,i2) ; integer i1,i2 ; begin
            w: = dark[i1]; dark[i1]: = dark[i2] ; dark[i2]: = w ;
            w: = dark2[i1] ; dark2[i1]: = dark2[i2] ; dark2[i2]: = w ;
            w: = k[i1] ; k[i1]: = k[i2] ; k[i2]: = w ;
            Bra: = kass[i1] ; kass[i1]: = kass[i2] ; kass[i2]: = Bra ;
            Bra: = posk[i1] ; posk[i1]: = posk[i2] ; posk[i2]: = Bra ;
            Bra = var[i1] ; var[i1]: = var[i2] ; var[i2]: = Bra ;
            for i: = 1 step 1 until Nak do begin
              w: = ak[i1,i] ; ak[i1,i]: = ak[i2,i] ; ak[i2,i]: = w end ;
            for i: = 1 step 1 until 20 do begin
              w = sk[i1,i] ; sk[i1,i]: = sk[i2,i] ; sk[i2,i]: = w end ;
            for i: = 1 step 1 until 20 do begin
              w = sk[i,i1] ; sk[i,i1]: = sk[i,i2] ; sk[i,i2]: = w end end Byta ;
procedure   Stuva ; begin integer Rn ; m: = 0 ;
            for ik: = Notvar + 1 step 1 until Nk do if kass[ik] then m: = m + 1 ;
            Nk: = Nk – m ; Rn: = Ntot ;
Ned:        Rn: = Rn – 1 ; if Rn = Notvar then goto Slut ;
            ik: = Rn ;
Upp:        if kass[ik] and not kass[ik + 1] then begin Byta(ik,ik + 1) ;
              if ik < Ntot – 1 then begin ik: = ik + 1 ; goto Upp end end ;
            goto Ned ;
Slut:       if kass[Nk + 1] then Klar: = true end Stuva ;


            goto Styre[Styr] ;
Styr1:      Klar: = false ; for ik: = 1 step 1 until Nk do var[ik]: = kass[ik]: = false ;
            input (60, 'N', Nvar) ;
            for i: = 1 step 1 until Nvar do begin input (60, 'N',ik) ; var[ik]: = true end ;
            Notvar: = 0 ;
            for ik: = 1 step 1 until Nk do if not var[ik] then begin Notvar: = Notvar + 1 ;
              if ik > Notvar then Byta(Notvar,ik) end ;
            input (60, 'N',Nin) ; Ntot: = Nk + Nin ;
            for i: = 1 step 1 until Nin do begin ik: = Nk + i ; input (60,'N',k[ik]) ;
              for j: = 1 step 1 until Nak do input (60, 'N',ak[ik,j]) ; input (60, 'N', w) ;
              dark[ik]: = – abs(w) ; var[ik]: = posk[ik]: = true ; kass[ik]: = false end ;
            Stuva ;
Styr2:      if Klar then goto Styrut ; Nk: = Nk + 1 ; i: = N: = 1 ; ivar[1]: = ik: = Nk ; STEKA ;
            kmin[1]: = 0 ; PLUSKA(1) ; if Orvar = 1 then Orvar: = – 2 ; STRECK ;
            output (61,'//2B'NU PROVAS' 2B'AK = ' 3B') ; AKUT ; Styr: = 3 ; goto LETA ;
Styr3:      Nge: = N: = Nk – Notvar ;
            for i: = 1 step 1 until N do ivarge[i]: = ivar[i]: = Notvar + i ;
            MINUT ; SIKIN ; Styr: = 4 ; Rskytt: = Nskytt + 1 ;
Styr4:      Rskytt: = Rskytt – 1 ; if Rskytt = 1 then Kassa: = true ;
            SKOTT ; for i: = 1 step 1 until N do begin ik: = ivar[i] ; STEKA end ;
            for i: = 1 step 1 until N do PLUSKA(i) ; goto LETA ;
Styr5:      Klar: = false ; if Ntot = Nk then goto Styrut ;
            m: = Nk ; j: = Ntot + 1;
Runt:       m: = m + 1 ; j: = j – 1 ; if j > m then begin Byta (m,j) ; goto Runt end ; goto Styr2 ;
Styr6:      stekfak: = 0.5 × stekfak ; Rskytt: = Nskytt + 1 ; Styr: = 4 ; goto Styr4 ;
Styr7:      Stuva ; N: = 0 ; Styr: = 8 ; Orvar: = 0 ; goto PROVIN ;
Styr8:      MINUT ; Kassdarr: = false ; if kass[Nk + 1] or Nk = Ntot then Klar: = true ;
            goto Styr2 ;
Styrut:     Styr: = 0 ; N: = Nk – Notvar ; STRECK ;
            for ik: = (Nk + 1) step 1 until Ntot do begin
              output (61, '/B'KASS',B–D.5D$_{10}$ + 3D2B,'DARR = ', – D.5D$_{10}$ + 3D2B, 'AK = '',
                      k[ik], dark2[ik]) ; AKUT ; kass[ik]: = false end ;
            SKRIK ; goto KNUT end STYRE ;
```

In this case, the new species is not accepted, and the earlier set is now combined with the next species in turn to be tested.

(*c*) The newcomer is accepted but one or more of the earlier species are thrown out by the F_σ condition (2).

After the program is finished with one parameter, *Styr* passes from 7 via 8 to 3, and a new parameter is added from the list (15). When all parameters from the list have been tested (*Klar* = **true**), the third group has disappeared, and after a final reordering by Stuva, *Styr*: = 0. The rejected parameters are printed, and also the final values for those accepted, with their standard deviations. The program goes to KNUT to find the new *Rurik*, and if it finds *Rurik* = 17, followed by (Input 5) above, a new cycle of STYRE is started, in which all the rejected parameters will have another chance.

The choice of the rejection factor F_σ = *sigfak*

The value for F_σ may be correlated to a value for the "level of confidence" which would mean the probability that (judging from these data) the corresponding parameter would be significantly positive:

F_σ	0	0.5	1	1.5	2	2.5	3
"confidence level" %	50	69	84	93.3	97.7	99.4	99.87

This is however an abstraction. The significance one really attaches to the results must depend on one's judgment of the experimental situation, and especially of the possibility of systematic errors. It is our experience that if the data are not very accurate, and cover a smaller concentration range than would be desirable, they can often accomodate a number of species which chemical intuition suggests to be unlikely. With such imperfect data one can at most hope to get some idea of the main species; then one should keep F_σ high, and restrict oneself to very simple models defined, say, "at most three species", or "at most two values for q".

On the other hand, with precise data over a wide concentration range, the choice of F_σ does not seem very critical. For instance, if one sets $F_\sigma = 1$ or 1.5, practically all the species that are accepted would also have passed the test with $F_\sigma = 3$ or 4.

The fact that a certain species has been rejected by the "species selector" does not mean that it does not exist. The only thing that can be stated is that the data used give no strong evidence for its existence. If one likes, one may give estimated "maximum" values for the formation constants of the rejected species, for instance log $(k + 3\sigma(k))$, as is done in many of our papers.

Choice of strategy

Examples of the application of MIKO and STYRE may be found in a number of recent papers from this Department, e.g. Hietanen and Sillén 1968, Kuča and Högfeldt 1968, Lundkvist 1968.

It seems advisable to start by calculating the "best" minimum with the set of major species indicated, for instance, by graphical treatment of the data, and then to add one new species after another. When computer time becomes less expensive, one might perhaps run through all conceivable (p, q) sets ranging e.g. from (1,1) to (20,20). At present, this is not feasible, for economic and other reasons, and one has to restrict oneself to a number of species that a) seem plausible or at least not impossible from the results of graphical treatment of the data, or b) have been suggested by other authors so that one thinks it is fair to test them whatever one thinks of their plausibility.

Once one has a set that gives a minimum in U, even if shallow, the procedure described above will give better and better U values. The order in which the species are tested may make a considerable difference in the amount of computer time required, but eventually it seems one will reach the same result in whichever order the species were added. The test for a "final" set of species is that no more species are accepted when the calculation is repeated with all rejected species. (*Rurik* = 17, *Styr* = 5, input 5.)

The program allows one to adjust the k_s together with the k while one is using the species selector. This is necessary in the treatment of spectrophotometric data where the k_s are molar absorptivities. It is a matter of judgment whether this facility should be used in other cases where the k_s are systematic errors. If the systematic errors are adjusted together with the k to minimize U, then some species may seem possible which did not meet the criterion (2) when systematic errors were assumed absent. On the other hand, a species that fulfilled the F_σ condition assuming the absence of systematic errors, may be eliminated when the errors are adjusted for. An example is given by Burkov, Lilič and Sillén, (1965). This is analogous to what may happen on adding a new species.

In general we prefer a conservative strategy; first those species are accepted that meet the F_σ condition with zero systematic errors, then we test if any of these may become uncertain if the systematic errors are adjusted together with k values for the remaining species.

ACKNOWLEDGEMENTS

We wish to thank Professor Nils Ingri and Dr. Erik Högfeldt for valuable discussions, and all our colleagues at this Department for a pleasant cooperation, and especially Ashley Stevens, B.A. (Oxon), for having revised the English of the manuscript. The present work is part of a series of investigations supported by Statens Naturvetenskapliga Forskningsråd (Swedish Natural Science Research Council). Computer time—mainly at CDC3600 in Uppsala—was made available by funds of the Royal Institute of Technology (KTH).

Department of inorganic chemistry, Royal Institute of Technology (KTH), S–100 44 Stockholm 70, Sweden

REFERENCES

Part 1 = SILLÉN, L. G., Acta Chem. Scand. *16*, 159 (1962).
Part 2 = INGRI, N., and SILLÉN, L. G., Acta Chem. Scand. *16*, 173 (1962).
Part 3 = SILLÉN, L. G., Acta Chem. Scand., *18*, 1085 (1964).
Part 4 = INGRI, N., and SILLÉN, L. G., Arkiv Kemi *23*, 97 (1964).
Part 5 = INGRI, N., KAKOŁOWICZ, W., SILLÉN, L. G., and WARNQVIST, B., Talanta *14*, 1261 (1967), *15*, XI (1968).
Part 6 = SILLÉN, L. G., and WARNQVIST, B., Arkiv Kemi *31*, 315 (1969).
BURKOV, K. A., LILIČ, L. S., and SILLÉN, L. G., Acta Chem. Scand. *19*, 14 (1965).
HIETANEN, S., and SILLÉN, L. G., Acta Chem. Scand. *22*, 265 (1968).
KUČA, L., and HÖGFELDT, E., Acta Chem. Scand. *22*, 183 (1968).
LUNDKVIST, M., Acta Chem. Scand. *22*, 281 (1968).

Tryckt den 26 augusti 1969

Uppsala 1969. Almqvist & Wiksells Boktryckeri AB