ARKIV FÖR KEMI Band 31 nr 25

4.68 12 13 Read 9 October 1968

High-speed computers as a supplement to graphical methods. 6

A strategy for two-level LETAGROP adjustment of common and "group" parameters. Some features that avoid divergence

By Lars Gunnar Sillén and Björn Warnqvist

ABSTRACT

A recent edition of the minimizing program LETAGROP is given and discussed. The common parameters k (such as equilibrium constants) are adjusted on an upper level, and for each set of common parameters, the group parameters k_s (such as analytical errors, E_0 values, or molar absorptivities for each wavelength) are adjusted on a lower level so as to minimize the error square sum U. This strategy was necessitated in the application of LETAGROP to spectrophotometric data but also proved preferable for other problems.

Other new features in the program, as compared with the edition in part 4, are the blocks ENSAM and SLUSS, which take care of bad initial values for the parameters, and the procedures Stegupp and Minskasteg which correct for the steps for variation having been chosen too small or too large.

Following parts will discuss the block MIKO for eliminating negative values for equilibrium constants, the "species selector" STYRE, and special programs for some specific chemical problems.

The program LETAGROP (Part 1–4) is devised to minimize a quantity U, typically an error square sum, which is a function of certain experimental data, and of a number of adjustable parameters. The parameters may be of two types, common parameters k (such as equilibrium constants) which are the same for all data considered, and "group parameters" k_s, such as analytical errors and E_0 values which are characteristic of only a certain part of the data used. In the example in part 4, an analytical error is treated as a group parameter; other examples will appear in parts 8–10.

In a certain problem, let N_k be the total number of adjustable common parameters, N_{ks} the number of adjustable group parameters in each group of data, N_s the number of groups, and N the number of parameters to be adjusted in the unit operation of adjustment, which we shall for brevity refer to as a "shot".

The shot

A *shot* involves the following set of calculations. At first U is calculated for $\frac{1}{2}(N+1)(N+2)$ parameter sets, defined by

$$\overset{\downarrow}{\mathbf{k}} = \overset{\downarrow}{\mathbf{c}} + \mathbf{SH}\overset{\downarrow}{\mathbf{v}} \qquad (1 = 3{:}4)$$

Here, $\vec{\mathbf{k}}$ is the vector of the parameters to be varied, $\vec{\mathbf{c}}$ is a central set of parameters, $\mathbf{H}$ is a diagonal step matrix, and $\mathbf{S}$ is a triangular "twist matrix" (part 3, p. 1088). Finally, $\vec{\mathbf{v}}$ is a variation vector chosen so that during the shot all elements are 0 except one or two at a time that are $+1$ or -1.

From these U values, the coefficients are calculated in the second-degree surface, typically a generalized elliptical paraboloid through these points:

$$U = U_c - 2\vec{\mathbf{p}}\overset{\downarrow}{\mathbf{v}} + \vec{\mathbf{v}}\mathbf{R}\overset{\downarrow}{\mathbf{v}} = U_0 + (\vec{\mathbf{v}} - \vec{\mathbf{v}}_0)\,\mathbf{R}(\overset{\downarrow}{\mathbf{v}} - \overset{\downarrow}{\mathbf{v}}_0) \qquad (2 = 3{:}7, 3{:}11)$$

One obtains directly the elements of $\vec{\mathbf{p}}$ and $\mathbf{R}$ (*pinne* and *ruta* in the program) and from $\vec{\mathbf{v}}_0 = \vec{\mathbf{p}}\mathbf{R}^{-1}$ one finds the position of the calculated minimum

$$\overset{\downarrow}{\mathbf{k}}_0 = \overset{\downarrow}{\mathbf{c}} + \mathbf{SH}\overset{\downarrow}{\mathbf{v}}_0 \qquad (3 = 3{:}13)$$

Then $U(\vec{\mathbf{k}}_0)$ is calculated, and if it is lower than earlier U values, it is accepted as the new central value. The twist matrix $\mathbf{S}$ is adjusted to minimize the non-diagonal terms in $\mathbf{R}$ in the next shot. The standard deviations $h_i\sigma(v_i)$ and $\sigma(k_i)$ are calculated, and stored (as *dark*[*ik*] and *dark2*[*ik*]), to be used in calculating the steps for the next shot ($h_i =$ *stek*[*i*] $=$ *stekfak* $\times$ *dark*[*ik*] in the program). The necessary equations are given in part 3, p. 1088–1092.

Adjustment of both k and k_s

When both common and group parameters are to be adjusted, we have followed three different strategies, which are listed here in chronological order:

Our *first strategy* (part 4) was to begin by keeping all the k_s constant and adjust the common parameters k, by shots including all data. Then the k were kept constant, and the k_s were adjusted by a number of shots for each group of data, using the same equations (1–3) except that for $\mathbf{S}$ we used simply a diagonal unit matrix. Then the cycle was repeated, adjusting first the k and then the k_s.

This strategy was sometimes adequate and converged to a minimum value of U after a few cycles. Sometimes, however, the convergence was slow.

Our *second strategy* was to adjust the k and k_s in the same shot. The k_s to be adjusted were then treated formally as $k[ik]$ with ik numbers $> N_k$. To take an example, in a case where one wished to adjust four k values and had six groups of data, in each of which two k_s values were to be adjusted, altogether $N = 4 + 2 \times 6 = 16$ parameters were to be varied and adjusted. Since each k_s influences only the part of U that comes from a single group, a certain economy could be achieved in the calculations, and the twist matrix terms s_{ij} could be set equal to 0 for all interactions between groups.

This strategy was faster than the first and often gave an acceptable minimum quite quickly. On the other hand, if some k_s was uncertain and did not influence the result too much, its uncertainty sometimes caused large deviations in the other parameters also.

We shall give no details of the second strategy since we think the third strategy is better. It was forced upon us by our wish to treat spectrophotometric data also; however, it proved preferable for other types of problems as well.

Example of spectrophotometric problem

Let us assume that the reagents A and B form a series of complexes A_pB_q, with formation constants β_{pq} so that

$$[A_pB_q] = c_{pq} = \beta_{pq}a^pb^q \tag{4}$$

where $a=[A]$, $b=[B]$. The mass balance conditions give for the total concentrations A and B

$$A = a + \Sigma pc_{pq};\ B = b + \Sigma qc_{pq} \tag{5}$$

In a certain solution, the absorptivity $E(\lambda)$ for some wavelength λ, and the molar absorptivity $\varepsilon(\lambda)$, calculated per B, may be expressed by

$$E(\lambda) = B\varepsilon(\lambda) = a\varepsilon_{10}(\lambda) + b\varepsilon_{01}(\lambda) + \Sigma c_{pq}\varepsilon_{pq}(\lambda) \tag{6}$$

provided Beer's law holds. Here, ε_{pq} are the molar absorptivities of the various complexes, ε_{10} and ε_{01} those of the component species.

The data usually include, for each one of a series of solutions, the values for $E(\lambda)$ for a certain number of selected wavelengths, and some information, such as (a, B) or (A, B) that allows one to calculate the individual concentrations a, b, c_{pq} once the equilibrium constants β_{pq} are assumed to be known.

The problem is to adjust the β_{pq} and the $\varepsilon_{pq}(\lambda)$ in such a way that the best fit is obtained with the data. As the error square sum to be minimized one may define e.g. $U = \Sigma(E_{\text{calc}} - E_{\text{exp}})^2$, the sum being taken over all solutions and wavelengths studied.

It seems natural to treat the β_{pq} as common parameters, and to treat the data for each separate λ as one group, with the $\varepsilon_{pq}(\lambda)$ as group parameters k_s, since the contribution from each wavelength is independent of whatever assumptions one makes about the ε_{pq} for other wavelengths.

To take a typical example, assume that we have extinction data for 12 wave lengths with 50 different solutions, and that we wish to test a model with five complexes, besides A and B. Then, for each λ we would have seven ε_{pq} values including ε_{10} and ε_{01} and even if the latter two were be determined by special experiments with pure A and B, we would have five ε_{pq} values to adjust. Hence, using the second strategy, we would have to adjust simultaneously $N = 5 + 5 \times 12 = 65$ parameters. Even if this calculation could be done, the rounding errors would cause a great inaccuracy, so the second strategy does not seem practical for this type of problem.

The third strategy, in brief, means that we adjust k and k_s by shots at two levels. As long as this is being done, the Boolean *Koks* is set **true**.

On the upper level (Boolean *Tage* = **false**), $\vec{\mathbf{k}}$ is varied systematically by means of (1). For each set $\vec{\mathbf{k}}$ of the common parameters, the k_s are adjusted at the lower level (Tage = **true**). By one or more shots for each separate group of data, one thus obtains the minimum possible contribution to U from that group, and the sum gives the lowest value of U attainable with that set $\vec{\mathbf{k}}$, which is carried to the higher level.

After the necessary number of points $U(\vec{\mathbf{k}})$ have been obtained, the position of the minimum $\vec{\mathbf{k}}_0$ is calculated on the upper level, using (2) and (3) as usual.

General plan

The present LETAGROP program, which involves the third strategy, has been so much improved, both in plan and in details, that we have found it advisable to give practically the full new ALGOL text in Table 1.

Three procedures for matrix inversion and multiplication have not been changed since part 4, the blocks MIKO and STYRE will be discussed in part 7, and special texts for UBBE and PUTS will be given in papers on special types of problem. In these cases Table 1 gives only a reference.

The input/output orders are made according to the ACM standard, but can easily be rewritten for a compiler with another preference.

For convenience with computers that use segmenting, the logical units have been made into real blocks. If the size limitations of the compiler necessitate compilation in parts, we suggest that INVERT, MULLE and PINUS and also if necessary MIKO and UBBE are compiled as external procedures.

The present program represents several years of experience with a variety of minimizing problems. We do not pretend that it could not be written more elegantly or efficiently nor that it could not misbehave in some cases we have not met with, but in our experience it works.

Function of blocks

Fig. 1 gives a block diagram of the present program, which may be compared with the earlier edition, Fig. 1 in part 4.

KNUT is the central point, to which the program returns after each task has been fulfilled. The nature of the next task is defined by a control number, *Rurik* which is also indicated in Fig. 1.

DATA reads and stores the original data (*ag*, *as*, *ap*) which will remain unchanged during the calculations; PUTS (connected with DATA) may calculate and store some useful derived quantities, often as new *ap*.

LÄSK reads preliminary values for the adjustable k and k_s, and also some connected constant quantities *ak*, such as the p, q values corresponding to a β_{pq}.

UBBE calculates U.

STEG reads orders for how to vary $\vec{\mathbf{k}}$ and $\mathbf{k}_s$ in the next shot, which is triggered off by putting $Rurik = 5$.

LETA varies $\vec{\mathbf{k}}$ (or $\mathbf{k}_s$) according to equation (1) and from the U values obtained calculates the elements of $\vec{\mathbf{p}}$ and $\mathbf{R}$ in (2). GROP calculates the new minimum $\vec{\mathbf{k}}_0$ by means of (3). MIKO (part 7) tries to find the best acceptable minimum when GROP has given negative values to some parameters (such as equilibrium constants) that are protected from being negative.

PROVA checks whether the calculated minimum is an improvement.

Table 1. Text of present edition of LETAGROP, with the exception of (a) three procedures given in Part 4, (b) the blocks MIKO and STYRE to be given and discussed in Part 7, and (c) the special blocks UBBE and PUTS to be given and discussed in connection with each separate type of problem. Capitals indicate labels and procedures belonging to the main block. A few minor improvements have been inserted in proof (March, 1969).

```
          begin comment Letagrop ;
          real darr1, darr2, det, ln10, loge, RoF, sigfak, sig2y, stegbyt, stekfak, tolU, U, U1,
                U1ko, U2, U2ko, Uc, Ucko, Uko, Umin, Uminko, Uno, w, w1, x, y ;
          integer Rurik, Orvar, arum, cell, i, ik, j, jk, m, N, Nak, Nap, Napa, Nas, Nge, Ngeko,
                Nido, Nidoko, Nk, Nko, Nkom, Nks, Nok, Notvar, Npunkt, Ns, Nskott,
                Nskytt, Ntot, Nx, Orvarko, Ri, Rido, Ridoko, Riko, Rj, Rjko, Rp, Rs, Rs1,
                Rs2, Rskott, Rskytt, Skrikut, Slusk, Sluskko, Styr, Typ, val, vmax ;
          boolean Bra, Fas2, Kassa, Kassdarr, Klar, Koks, Mink, Minkko, Poskis, Prov,
                Provko, Rakt, Tage ;
          real array ag[1:10], ak[1:20,1:12], ap[1:6000], as[1:25,1:12], dark, dark2[1:20], darks,
                darks2[1:25,1:14], k[1:20], kbom, kc, kcko, kmin, kminko[1:16], ks,
                ksmin[1:25,1:14], kv, kvko, pinne, pinneko[1:20], ruta, rutako, s, SH, SHko,
                sko[1:16,1:16], sk[1:20,1:20], start[1:5], stek, stekko[1:16], tol[1:5], v,
                vbom[1:16] ;
          integer array apcell[1:25], ido, idoko[1:5], ivar, ivarge, ivarko, ivargeko[1:16], Niks
                [1:25], Np[1:25], pot[1:16], Vaks[1:25,1:10] ;
          boolean array kass, posk, var[1:20] ;
procedure AKUT ; begin integer i ;
          for i: = 1 step 1 until Nak do output (61,'-3ZD.2D',ak[ik,i]) end AKUT ;
procedure DARRA(i); value i ; integer i ; begin
          if Tage then begin darks[Rs,ik]: = darr1 ; darks2[Rs,ik]: = darr2 ;
            if Skrikut > -1 then output (61, '/2B'SATS',ZD,2B,'KS',ZD,' = ',B-D.5D$_{10}$+3D,
                 'DARR = ',2B-D.3D$_{10}$+3DB, -D.3D$_{10}$+3D',Rs,ik,kbom[i],darr1,darr2) end
            else begin dark[ik]: = darr1 ; dark2[ik]: = darr2 ;
              output (61,'/2B'K',2ZD,' = ',B-D.5D$_{10}$+3D2B,'DARR1 = ',B--D.3D$_{10}$+3D2B,
                 'DARR2 = ', -D.3D$_{10}$+3D2B,'AK = '',ik,kbom[i],darr1,darr2) ; AKUT
          end end DARRA ;
procedure INVERT = part 4
procedure MININ ; begin Uc: = U: = Umin ;
          for i: = 1 step 1 until N do begin ik: = ivar[i]; kv[i]: = kc[i]: = kmin[i];
            if Tage then ks[Rs,ik]: = kmin[i] else k[ik]: = kmin[i] end;
          if Koks and not Tage then for Rs: = Rs1 step 1 until Rs2 do
            for ik: = 1 step 1 until Nks do ks[Rs,ik]: = ksmin[Rs,ik] end MININ ;
procedure MINUT ; begin integer ik; Umin: = U ;
          for i: = 1 step 1 until N do begin ik: = ivar[i] ;
            kmin[i]: = if Tage then ks[Rs,ik] else k[ik] end ;
          if Koks and not Tage then for Rs: = Rs1 step 1 until Rs2 do
            for ik: = 1 step 1 until Nks do ksmin[Rs,ik]: = ks[Rs,ik] end MINUT ;
procedure MULLE = part 4
procedure PINUS = part 4
procedure PLUSKA(i) ; value i ; integer i ; begin real max ; ik: = ivar[i] ;
          if not posk[ik] then goto Slut ; max: = 0 ;
          for j: = 1 step 1 until N do max: = max + abs(s[i,j] × stek[j]) ;
          if max > k[ik] then begin k[ik]: = max ;
            if Orvar = 1 then Orvar: = -2 ; if Orvar > 1 or Orvar = -3 then Orvar: = -5 end;
Slut:     kv[i]: = kc[i]: = k[ik] end PLUSKA ;
procedure PROVAR ; begin if Tage then output (61, '/B'SATS',2DB,'PROVA'',Rs)
                else output (61, '/2B'PROVA'8B') ;
          for i: = 1 step 1 until N do
            output (61,'10BZD',ivar[i]) end PROVAR ;
procedure SATSUT ; begin
          output(61,'/B'SATS',ZD',Rs) ; output(61,'/B 'AS = '') ;
```

```
             for j: = 1 step 1 until Nas do output(61,'B – D.5D_{10} + 3D',as[Rs,j]) ;
                     output(61,'/B'KS = '') ;
             for j: = 1 step j until Nks do output(61, 'B – D.5D_{10} + 3D',ks[Rs,j])
             end SATSUT ;
procedure    SIGGE ; begin real sigy ;
             if Uno < 0 then begin sig2y: = – 1 ; output (61, '/'MINUSGROP'') end ;
               else begin m: = if Tage then Np[Rs] else Npunkt ;
                 if Koks and not Tage then for Rs: = Rs1 step 1 until Rs2 do m: = m – Niks[Rs];
                 if m – N ≤ 0 then begin sig2y: = – 1 ; output(61, '/'PUNKTBRIST'') ; goto Slut
                     end ;
               sig2y: = Uno/(m – N) ; sigy: = sqrt(sig2y) ;
               if not (Tage and Skrikut < 0) then output(61, '/B'SIGY = ',3ZD.5D',sigy) end ;
Slut:        end SIGGE ;
procedure    SIKIN ; begin
             for i: = 1 step 1 until N do begin ik: = ivar[i] :
               for j: = 1 step 1 until N do begin jk: = ivar[j] ;
                 if i > j then s[i,j]: = 0 else s[i,j]: = sk[ik,jk] end end end SIKIN ;
procedure    SIKREN(m) ; integer m ; begin
             for i: = 1 step 1 until m do for j: = 1 step 1 until m do s[i,j]: = 0 ;
             for i: = 1 step 1 until m do s[i,i]: = 1 end SIKREN ;
procedure    SKOTT ; begin
             if Tage then output (61, '//,2B'SATS',ZD,2B,'SKOTT'',Rs)
               else output(61,'//,10B SKOTT'') ;
             for i: = 1 step 1 until N do output(61,'10BZD',ivar[i]) end SKOTT ;
procedure    SKRIK ; begin
             output (61,'// B'K(IK) = '') ;
             for ik: = 1 step 1 until Nk do begin output (61,'/B'K',ZD,' = ',B – D.5D_{10} + 3D,3B,
                     'DARR = ',B – D.3D_{10} + 3D,3B,'AK = '',ik,k[ik],dark2[ik]) ; AKUT end ;
             if Nks > 0 then for Rs: = 1 step 1 until Ns do begin output (61, '/'SATS',ZD',Rs) ;
               for i: = 1 step 1 until Nks do begin if i = 4 then output(61,'/6B') ; output(61, ' 2B,
                     'KS',ZD,' = ',B – D.5D_{10} + 3DB, 'DARR = ',B – D.3D_{10} + 3D', i, ks[Rs,i],
                     darks2[Rs,i])end end ;
             STRECK end SKRIK ;
procedure    STEKA ; begin w: = if Tage then darks[Rs,ik] else dark[ik] ;
             stek[i]: = if w > 0 then stekfak × w else – w end STEKA ;
procedure    STRECK ; begin output(61, '/');
             for m: = 1 step 1 until 50 do output(61,'B' – '') ; output(61, '/') end STRECK ;
procedure    UVAR ; begin
             if Tage then begin output (61, '/B 'U = ',B – D.6D_{10} + 3DB, 'KS = '',U) ;
               for i: = 1 step 1 until N do output (61,'B – D.3D_{10} + 3D',ks[Rs, ivar[i]]) end
             else begin output (61, '/B 'U = ', B–D.6D_{10} + 3DB, 'KV = '', U) ;
               for i: = 1 step 1 until N do output (61, 'B–D.3D_{10} + 3D', k[ivar[i]]) end end UVAR ;


             ln10: = ln(10) ; loge: = 1/ln10 ; N: = Nido: = Rs: = Slusk: = Styr: = 0 ;
             Fas2: = Kassa: = Kassdarr: = Koks: = Mink: = Prov: = Rakt: = Tage: = false  ;
             Skrikut: = 1 ; stekfak: = 0.5 ; vmax: = 120; RoF: = 0.086167 ; tolU: = 0.000001 ;
             output(61, '/'LETAGROP'') ; goto KNUT ;
DATA:        begin integer Nag;
             if Rs = 0 and not Klar then begin input (60, '4(N)', Ns,Nag,Nas,Nap) ; Napa: = Nap ;
                     output (61, '/B,'AG = '') ;
               for i: = 1 step 1 until Nag do begin input (60,'N',ag[i]); output (61,'B – D.5D_{10} +
                     3D',ag[i])end ;
               Klar: = true ; goto PUTS end ;
             Rs: = Rs + 1 ;
             if Rs > Ns then begin Rs1: = 1; Rs2: = Ns; Npunkt: = 0 ;
               for Rs: = Rs1 step 1 until Rs2 do Npunkt: = Npunkt + Np[Rs]  ;
               output (61,'/B'SATS', 2ZDB,' – '2ZD3B, 3ZDB,'PUNKTER'', Rs1, Rs2, Npunkt) ;
               goto KNUT end;
             input (60,'N', Np[Rs]) ;
             cell: = apcell[Rs]: = if Rs = 1 then 0 else cell ;
             for i: = 1 step 1 until Nas do input (60,'N',as[Rs,i]) ;
```

```
          for Rp: = 1 step 1 until Np[Rs] do begin for i: = 1 step 1 until Nap do input (60,'N',
                    ap[cell + i]) ; cell: = cell + Napa end ;
          goto PUTS end DATA;
ENSAM:    begin switch Ens: = Ens1, Ens2, Ens3, Ens4, Ens5, Ens6, Ens7, Ens8, Ens9, Ens10;
          if (Orvar = 1 or Umin > U) and not Prov then MINUT ;
          ik: = ivar[1] ; if Orvar = − 1 then Orvar: = 1 ; goto Ens[Orvar] ;
Ens1:     Uc: = U ; kv[1]: = kc[1] + stek[1] ; Orvar: = 2 ; goto Ut ;
Ens2:     U1: = U ; kv[1]: = kc[1] − stek[1] ; Orvar: = 3 ; goto Ut ;
Ens3:     U2: = U ; goto Gropen ;
Ens4:     Prov: = false ; if U ⩽ Umin + Umin × tolU or (Tage and Poskis) then goto Träffen ;
          Mink: = false ; if Uc > U2 then goto Ens11 ; if Uc > U1 then goto Ens12 ;
          stek[1]: = 0.5 × stek[1] ; kv[1]: = kc[1] + stek[1] ; Orvar: = 6 ; goto Ut ;
Ens5:     U1: = U ; goto Gropen ;
Ens6:     if U < Uc then begin U2: = Uc ; Uc: = U ; kc[1]: = kv[1] ; goto Gropen end ;
          U1: = U ; kv[1]: = kc[1] − stek[1] ; Orvar: = 7 ; goto Ut ;
Ens7:     if U < Uc then begin U1: = Uc ; Uc: = U ; kc[1]: = kv[1] end else U2: = U ;
          if (U1 + U2 − 2 × Uc) < 21 × tol U × Uc then begin MININ ; goto Träffen end ;
          goto Gropen ;
Ens8:     U2: = U ; kv[1]: = kc[1] ; Orvar: = 9 ; goto Ut ;
Ens9:     Uc: = U ; goto Gropen ;
Ens10:    Uc: = U ; kv[1]: = kc[1] + stek[1] ; Orvar: = 5 ; goto Ut ;
Ens11:    if posk[ik] and kc[1] < 3 × stek[1] and not Tage then goto Ens13 ;
          Uc: = U2 ; kc[1]: = kc[1] − stek[1] ; stek[1]: = 2 × stek[1] ; kv[1]: = kc[1] − stek[1] ;
          Orvar: = 3 ; goto Ut ;
Ens12:    Uc: = U1 ; kc[1]: = kc[1] + stek[1] ; stek[1]: = 2 × stek[1] ; kv[1]: = kc[1] + stek[1] ;
          Orvar: = 5 ; goto Ut ;
Ens13:    stek[1]: = kc[1]: = 0.5 × kc[1] ; U1: = Uc ; kv[1]: = 0 ; Orvar: = 8; goto Ut ;
Ens14:    if U1 < (U2 + w1) then goto Kasso ;
          stek[1]: = kv[1]: = kc[1]: = 0.5 × kc[1] ; U1: = Uc ; goto Ut ;
Gropen:   X: = 0.5 × (U1 + U2) − Uc ; w1: = 10 × Uc × tol U ;
          if X < w1 then begin if Orvar = 9 then goto Ens14 ; if U1 > (U2 + w1) then goto Ens11
                    else goto Ens12 end ;
          vbom[1]: = − 0.25 + (U1 − U2)/X ; kbom[1]: = kc[1] + stek[1] × vbom[1] ; Uno: = Uc
                    − X × vbom[1]↑2 ;
          SIGGE ;
          if sig2y > 0 then darr1: = darr2: = stek[1] × sqrt(sig2y/X)
             else darr1: = darr2: = − abs (stek[1]) ;
          DARRA(1) ;
          if not Tage and posk[ik] and kbom[1] < 0 then begin kv[1]: = 0 ; output (61, '/2B
                    'MIKO'') ;
             if not Mink then Mink: = true else Mink: = false ;
             if Orvar = 9 or not Mink then begin U: = U2 ; k[ik]: = kv[1] ; goto Ens4 end end
          else begin kv[1]: = kbom[1] ; Mink: = false end ;
          if Tage and Poskis and kv[1] < 0 then kv[1]: = 0 ;
          if not (Koks and Skrikut < 0) then PROVAR ;
          Orvar: = 4 ; Prov: = true ; goto Ut ;
Träffen:  MINUT ;
          if Mink then begin U2: = U ;
             stek[1]: = 0.5 × dark[ik] ;
             kv[1]: = kc[1]: = stek[1] ; Orvar: = 10 ; goto Ut end ;
          Orvar: = 1 ; Uc: = U ; kc[1]: = kv[1] ;
          if Kassa and not Tage then begin Styr: = 8 ; goto STYRE end ;
          goto PROVUT ;
Ut:       if Tage then ks[Rs,ik]: = kv[1] else k[ik]: = kv[1]; goto UBBE ;
Kasso:    output (61,'/'KONVEX TILL NOLL'') ; Uno: = Umin; SIGGE; darr1: = darr2: =
                    − 2 × kc[1] × sig2y/(U1 − U2) ; DARRA(1) ; Mink: = false ; Orvar: = 1;
                    MININ ; goto PROVUT
          end ENSAM ;

GROP:     begin integer Rv ; real array dia1, dia2[1:16], rut1, rutinv, vri[1:16,1:16] ;
          for i: = 1 step 1 until N do for j: = 1 step 1 until N do rutinv[i,j]: = ruta[i,j] ;
          INVERT(N,rutinv,1_{10} − 35, det, SING) ;
          PINUS(pinne, rutinv, N, vbom, 1) ;
          Uno: = Uc ; for i: = 1 step 1 until N do Uno: = Uno − pinne[i] × vbom[i] ;
          SIGGE ; if not (Tage and Skrikut < 0) then output (61,'/2B 'KBOM = '') ;
          PINUS(vbom, SH, N, kbom, − 1) ; MULLE(SH, rutinv, N, N, N, rut1, 1) ;
          for i: = 1 step 1 until N do begin kbom[i]: = kbom[i] + kc[i] ; dia1[i]: = rutinv[i,i] ;
            dia2[i]: = 0 ;
            for m: = 1 step 1 until N do dia2[i]: = dia2[i] + rut1[i,m] × SH[i,m] end ;
          for i: = 1 step 1 until N do begin ik: = ivar[i] ;
            if Tage and not Poskis then ks[Rs,ik]: = kbom[i] ;
            if dia1[i] > 0 and Uno > 0 then darr1: = abs(sqrt(sig2y × dia1[i]) × SH[i,i])
              else darr1: = − abs(stek[i]) ;
            if dia2[i] > 0 and Uno > 0 then darr2: = sqrt(sig2y × dia2[i])
              else darr2: = − abs(stek[i]) ;
            DARRA(i) end ;
Vrid:     for i: = 1 step 1 until N do for j: = 1 step 1 until N do vri[i,j]: = 0 ;
          for i: = 1 step 1 until N do vri[i,i]: = 1 ;
          Rv: = N ;
Vänd:     Rv: = Rv − 1 ;
          for i: = 1 step 1 until Rv do for j: = 1 step 1 until Rv do rutinv[i,j]: = ruta[i,j];
          INVERT(Rv,rutinv,1_{10} − 30,det,SING) ;
          for i: = 1 step 1 until Rv do begin w: = 0 ;
            for m: = 1 step 1 until Rv do w: = w − rutinv[i,m] × ruta[m,Rv + 1] ; vri[i,Rv + 1]: = w
                    end ;
          if Rv > 2 then goto Vänd ;
          vri[1,2]: = − ruta[1,2]/ruta[1,1] ;
          for i: = 1 step 1 until N do for j: = 1 step 1 until N do rut1[i,j]: = vri[i,j]/SH[j,j] ;
          MULLE(SH,rut1,N,N,N,s,1) ; m: = 0 ;
          if Tage then begin if Poskis then goto MIKO else goto PROVIN end ;
          if Skrikut > − 1 or (Skrikut = − 1 and Koks) then output(61, '/B'SIK'') ;
          for i: = 1 step 1 until N do for j: = i + 1 step 1 until N do begin
            ik: = ivar[i] ; jk: = ivar[j] ; sk[ik,jk]: = s[i,j] ;
            if Skrikut > − 1 or (Skrikut = − 1 and Koks) then begin m: = m + 1 ;
              if m = 6 then begin m: = 1 ; output(61, '/4B') end ;
              output(61,'2B,2ZD,2ZD,B − D.2D_{10} + 3D', ik,jk,sk[ik,jk]) end end ;
          goto MIKO end GROP ;
KNUT:     begin switch Knutte: = Rur1, Rur2, Rur3, Rur4, Rur5, Rur6, Rur7, Rur8, Rur9,
                    Rur10, Rur11, Rur12, Rur13, Rur14, Rur15, Rur16, Rur17, Rur18, Rur19,
                    Rur20 ;
procedure Logkik(ik) ; integer ik ; begin x: = k[ik] ; y: = dark2[ik] ;
          if y ≤ 0 then output (61, '/B 'K', ZD, 'SAKNAR DARR'',ik)
          else if y < 0.2 × x then begin w: = ln(x) × loge + ak[ik,1]; w1: = 1.5 × (ln(x + y) − ln(x − y))
                    × loge ; output (61, '/B 'LOG K',ZD,B ' = ', − 2ZD.4D2B, ' + − ',2B − ZD.
                    4D',ik,w,w1) end
          else begin w: = x + 3 × y ;
            if w < 0 then output (61, '/B 'K',ZD,'EJ POS'', ik)
            else begin w1: = ln(w) × loge + ak[ik,1] ;
              if x > 0 then begin w: = ln(x) × loge + ak[ik,1] ; output (61, '/B 'LOG K',ZD2B,'
                  = ', − 2ZD.4D2B,'MAX = ', − 2ZD.4D',ik,w,w1) end
              else output (61,'/B 'K',ZD2B,'MAX = ', − 2ZD.4D', ik,w1) end
          end end Logkik;

          input (60,'N', Rurik) ;
          if Rurik ≠ 14 then output (61, '/B, 'RURIK = ', − ZD',Rurik) ;
          if Rurik < 1 or Rurik > 20 then goto FINAL ;
          goto Knutte[Rurik] ;
Rur1:     if Koks and Orvar = 1 then begin Rakt: = true;Nge: = N end; N: = 0; goto LETA ;
Rur2:     STRECK; output (61,'// 10B'UTTÅG'') ; goto Rur1 ;
Rur3:     STRECK ; Koks: = false ; goto STEG ;
Rur4:     input (60,'N', w) ; if w > 0 then stekfak: = w else tolU: = − w ;
          output (61,'/,'STEKFAK = ',ZD.3D2B, 'TOLU = '3D.D_{10} − 2D', stekfak, tolU) ;
          goto KNUT ;
```

```
Rur5:       for i: = 1 step 1 until N do begin ik: = ivar[i] ; STEKA end ; SIKIN ;
            for i: = 1 step 1 until N do PLUSKA(i) ; SKOTT ; goto LETA ;
Rur6:       Rs: = 0 ; Orvar: = 0 ; Klar: = false ; goto DATA ;
Rur7:       Orvar: = 0 ; goto LÄSK ;
Rur8:       input (60,'2(N)',Nok, stegbyt) ;
            for i: = 1 step 1 until Nok do input (60,'2(N)',start[i],tol[i]);
            output (61,'/, 'STEGBYT = ', ZD.3D2B, 'TOL = ', 5(2BD.D10 − 2D)', stegbyt,
                for i: = 1 step 1 until Nok do tol[i]); goto KNUT ;
Rur9:       input (60,'N',Typ) ; val: = 1 ; Koks: = false ; Orvar: = 0 ; Poskis: = false ;
            output (61,'/ 'TYP = ',ZD',Typ) ; output (61,'5B 'VAL = ',ZD',val) ;
            goto KNUT ;
Rur10:      input (60,'N',vmax) ; goto KNUT ;
Rur11:      input (60,'2(N)',Rs1,Rs2) ; Koks: = false ; Orvar: = 0 ; Npunkt: = 0 ;
            for Rs: = Rs1 step 1 until Rs2 do Npunkt: = Npunkt + Np[Rs] ;
            output (61,'/ 'SATS',2ZDB,' − ',2ZD,3B2ZDB,'PUNKTER'',Rs1,Rs2,Npunkt) ;
            goto KNUT ;
Rur12:      input (60,'N', Skrikut) ; output (61,'/,'SKRIKUT = ' − 2D', Skrikut) ; goto KNUT ;
Rur13:      SKRIK ; goto KNUT ;
Rur14:      output (61,'/') ; input (60, '/') ;
            for i: = 1 step 1 until 80 do begin input (60,'A',m);output (61,'A',m) end ; goto KNUT ;
Rur15:      output (61,'/') ;
            for ik: = 1 step 1 until Nk do if posk[ik] then Logkik(ik) ; goto KNUT ;
Rur16:      output (61,'↑') ; goto KNUT ;
Rur17:      input (60,'3(N)',Styr,sigfak,Nskytt) ; STRECK ;
            output (61,'/15B 'STYR = ',ZD3B,'SIGFAK = ',4ZD.D',Styr,sigfak) ; goto STYRE ;
Rur18:      input (60,'N',val) ; STRECK ; output (61,'/ 'VAL = ',ZD',val) ; Orvar: = 0 ;
            goto KNUT ;
Rur19:
Rur20:      Koks: = true ; Orvar: = 0 ; STRECK ; output (61, '/22B'KOKS'') ; goto STEG
            end KNUT ;
LETA:       if N = 0 then goto UBBE ; if Orvar = 0 or Orvar = − 2 then goto UBBE ; if N = 1 then
                goto ENSAM ;
            begin switch Letax: = Leta1, Leta2, Leta3, Leta4 ;
procedure   Komner ; begin real max, slask, Umi; integer jmax;
            Bra: = true;Umi: = if Nido = 0 then Umin else Uc ;
            if sig2y > 0 then begin x: = (U − Umi)/sig2y ;
              if x > 2 then begin w: = 0.7/sqrt(x) ; Bra: = false end end ;
            if sig2y < 0 then begin x: = U/Umi − 1 ;
              if x > 0.04 then begin w: = 0.1/sqrt(x) ; Bra: = false end end ;
            if Bra then goto Slut ; if Skrikut > − 2 then output (61, '/8B 'KOM NER'') ;
            for m: = 1 step 1 until N do begin ik: = ivar[m] ;
              if kmin[m] < k[ik] then begin max: = 0 ;
                for j: = 1 step 1 until N do begin slask: = abs(s[m,j] × stek[j]) ;
                  if slask > max then begin max: = slask ; jmax: = j end end ;
                if jmax = m then stek[m]: = w × stek[m] else s[m,jmax]: = w × s[m,jmax] end end;
            MININ ; for i: = 1 step 1 until N do PLUSKA(i) ;
Slut:       end Komner ;
procedure   Minskasteg ; begin
            if Skrikut > − 1 or (Skrikut = − 1 and Koks and not Tage) then output (61,'/'MINSKA
                STEG', 2ZD', ik) ;
            w: = sqrt(Uc/(w1 × m)) ; stek[Ri]: = w × stek[Ri] ; SHber ; Ri: = Ri − 1 ; goto Uppi end
                Minskasteg ;
procedure   SHber ; begin for i: = 1 step 1 until N do for j: = 1 step 1 until N do SH[i,j]: = s[i,j] ×
                stek[j] end SHber ;
procedure   Stegupp ; begin
            if Skrikut > − 1 or(Skrikut = − 1 and Koks and not Tage) then output (61, '/ 'STEG
                UPP',2ZD',ik) ;
            if stek[Ri] = 0 then stek[Ri]: = k[ik] × 0.001 else stek[Ri]: = 10 × stek[Ri] ;
            if not Tage then for i: = 1 step 1 until N do
              begin ik: = ivar[i] ; k[ik]: = kc[i] ; PLUSKA(i) end ;
            SHber ; Ri: = Ri − 1 ; if Orvar = − 5 then goto UBBE ; if Orvar ≠ − 3 then goto Uppi
            end Stegupp ;
```

```
                  for i: = 1 step 1 until N do v[i]: = 0 ;
                  if Orvar = 1 then goto Leta0 ;
                  if Orvar = −5 then begin Komner; if not Bra then goto UBBE; SHber; Uc: = U ; goto
                          Uppi end ;
                  if Orvar = −4 then goto Leta2 ;
                  if Orvar = −3 then goto Leta3 ;
                  if Orvar = −1 then begin Komner ; if not Bra then begin Orvar: = −2 ; goto  UBBE
                          end ; Orvar: = 1 end ;
                  if U > Umin then goto Letax[Orvar] ;
                  if Tage and Poskis then for i: = 1 step 1 until N do begin ik: = ivar[i] ;
                     if ks[Rs,ik] < 0 then goto Letax[Orvar] end ;
Leta0:            MINUT ;
                  if Slusk = 0 then begin
                     if Orvar = 1 then Nido: = 0 ;
                     if Orvar = 2 or Orvar = 3 then begin Nido: = 1 ; ido[1]: = Ri end ;
                     if Orvar = 4 then begin Nido: = 2 ; ido[1]: = Rj; ido[2]: = Ri end end ;
                  if not Tage then sig2y: = U/Npunkt ; comment for Komner ;
                  goto Letax[Orvar] ;
Leta1:            Uc: = U ; for i: = 1 step 1 until N do for j: = 1 step 1 until N do ruta[i,j]: = 0 ;
                  SHber ; Ri: = 0 ; goto Uppi ;
Leta2:            U1: = U ; v[Ri]: = −1 ; if Orvar = −4 then Orvar: = −3 else Orvar: = 3 ; goto Kvber ;
Leta3:            U2: = U ;
                  m: = if Tage then Np[Rs] else Npunkt ; ik: = ivar[Ri] ; w1: = 0.5 × (U2 + U1) − Uc  ;
                  if w1 × m > 2 × Uc then Minskasteg ;
                  w: = if tolU < 0.0001 then 10 × tol U × Uc else 0.001 × Uc ;
                  if abs(U1 − Uc) < w and abs(U2 − Uc) < w then Stegupp ;
                  if Orvar = −3 then begin U: = Uc ; goto Leta1 end ;
                  if U2 < U1 then begin U2: = U1 ; U1: = U; stek[Ri]: = −stek[Ri]; SHber end ;
                  pinne[Ri]: = 0.25 × (U2 − U1) ; ruta[Ri,Ri]: = w1 ; Orvar: = 4 ; goto Jupp ;
Leta4:            w: = 0.5 × (U − Uc) + pinne[Ri] + pinne[Rj] − 0.5 × (ruta[Ri,Ri] + ruta[Rj,Rj]) ;
                  ruta[Ri,Rj]: = ruta[Rj,Ri]: = w ; goto Jupp ;
Jupp:             Rj: = Rj + 1 ; if Rj ≥ Ri then goto Uppi ;
                  v[Ri]: = 1 ; v[Rj]: = 1 ; goto Kvber ;
Uppi:             Ri: = Ri + 1 ; if Ri > N then goto Letaslut ; Rj: = 0 ; v[Ri]: = 1 ;
                  if Orvar = −5 then Orvar: = −4 else Orvar: = 2 ;
Kvber:            PINUS(v,SH,N,kv, −1) ;
                  for i: = 1 step 1 until N do begin ik: = ivar[i] ; kv[i]: = kv[i] + kc[i] ;
                     if Tage then ks[Rs,ik]: = kv[i] else k[ik]: = kv[i] end ; goto UBBE ;
Letaslut:         Ri: = 0 ; Rj: = 0 ; Orvar: = 1 ; goto GROP end LETA ;
LÄSK:             begin integer Nbyk, Nbyks, Negk, skin ;
procedure         Darkin ; begin dark[ik]: = dark2[ik]: = −1 ; posk[ik]: = true end Darkin ;
procedure         Darksin ; begin darks[Rs,ik]: = darks2[Rs,ik]: = −1 end Darksin ;
                  input (60,‘2(N)’, m,Nbyk) ;
                  if Nbyk = −1 then begin input (60, ‘N’,Negk) ;
                     for i: = 1 step 1 until Negk do begin input (60, ‘N’,ik) ; posk[ik]: = false end ;
                     goto KNUT end ;
                  Nk: = m ;
                  if Nbyk < Nk then for m: = 1 step 1 until Nbyk do begin input (60, ‘2(N)’,ik,k[ik]) ;
                     for i: = 1 step 1 until Nak do input (60,‘N’,ak[ik,i]); Darkin end ;
                  if Nbyk = Nk then begin input (60, ‘N’, Nak) ;
                     for ik: = 1 step 1 until Nk do begin input (60, ‘N’,k[ik]) ;
                        for i: = 1 step 1 until Nak do input (60, ‘N’,ak[ik,i]) end ;
                     for ik: = 1 step 1 until 20 do for jk: = 1 step 1 until 20 do sk[ik,jk]: = 0 ;
                     for ik: = 1 step 1 until 20 do begin sk[ik,ik]: = 1 ; Darkin end end ;
                  input (60,‘2(N)’,Nks,Nbyks) ;
                  if Nbyks = −1 then begin input (60, ‘N’,m) ;
                     for i: = 1 step 1 until m do begin input (60, ‘N’,ik)  ; for Rs: = 1 step 1 until Ns do
                        begin ks[Rs,ik]: = 0 ; Darksin end end end ;
                  if Nbyks = 0 and Nks ≠ 0 then begin input (60, ‘N’, m) ;
                     for i: = 1 step 1 until m do input (60, ‘N’, Niks [i]);
                     for Rs: = 1 step 1 until Ns do begin
                        for ik: = 1 step 1 until Nks do begin ks[Rs,ik]: = 0; Darksin end ;
```

```
                for i: = 1 step 1 until m do input (60, 'N', ks[Rs,Niks[i]) end end;
            if Nbyks = Nks then begin for Rs: = 1 step 1 until Ns do for ik: = 1 step 1 until Nks do
                    begin input (60, 'N',ks[Rs,ik]); Darksin end end
              else if Nbyks = 1 then begin input (60, 'N',ik) ;
                for Rs: = 1 step 1 until Ns do begin input (60, 'N',ks[Rs,ik]); Darksin end end ;
            input (60, 'N',skin) ; for m: = 1 step 1 until skin do input (60,'3(N)',ik,jk, sk[ik,jk]) ;
            SKRIK ; goto KNUT end LÄSK ;
MIKO:        = part 7
SING:       output (61, '///B'SING'///') ; goto KNUT ;
PROVIN:     Prov: = true ; if not (Koks and Skrikut < 0) then PROVAR ; goto UBBE ;
PROVA:      Prov: = false ; if Kassdarr and not Tage then goto STYRE ;
            if U ⩽ Umin or(Tage and Poskis) then goto Träff ;
            if Nido = 0 then begin if not (Tage  and  Skrikut < 0)  then  output(61,  '/'GAMLA
                    KONSTANTER''); Slusk: = 0 end
            else begin  if Slusk = 0 then begin Slusk: = 1 ; Rido: = 0; Nge: = N ; for i: = 1 step 1
                    until Nge do ivarge[i]: = ivar[i] end ;
              if not (Tage and Skrikut = - 2) then begin output (61,'/'SLUMPSKOTT'') ;
                if Slusk = 1 then for i: = 1 step 1 until Nido do output (61,'2ZD',ivar[ido[i]]);
                    output(61, '/'UMIN = ',BD.6D$_{10}$ + 3DB, 'KMIN = '',Umin) ;
                for i: = 1 step 1 until N do output (61,'B - D.3D$_{10}$ + 3D',kmin[i]) end end ;
            MININ ; goto PROVUT ;
Träff:      MINUT ; for i: = 1 step 1 until N do kc[i]: = kmin[i] ; Uc: = Umin ;
            if Slusk = 0 then Nido: = 0 ;
            if Rakt then begin Rakt: = false ; N: = Nge end ;
PROVUT:     if Slusk > 0 then goto SLUSS ;
PROVUT1:    if Tage then goto SÄRK ; if Styr > 0 then goto STYRE ; goto KNUT ;
SLUSS:      begin switch Sluss: = Slusk1, Slusk2, Slusk3, Slusk4 ;
procedure   Sluska (L, Rido) ; integer L, Rido ; begin i: = L ;
            m: = ido[Rido] ; ik: = ivar[i]: = ivarge[m] ; STEKA ;
            if Tage then kc[i]: = kmin[i]: = ks[Rs,ik]
              else begin kmin[i]: = k[ik] ; PLUSKA(i) end end Sluska ;
            goto Sluss[Slusk] ;
Slusk1:     Rido: = Rido + 1 ;
            if Rido > Nido then begin if Nido = 1 then goto Slusk4 else goto Slusk2 end ;
            N: = 1 ; Sluska(1,Rido) ; output (61, '/') ; goto LETA ;
Slusk2:     N: = 2 ; Sluska(1,1) ; Sluska(2,2) ; SIKREN(2) ; output (61,'/'); Slusk: = 3 ; goto LETA ;
Slusk3:     Sluska(1,1) ; Sluska(2,2) ; Slusk: = 4 ; goto LETA ;
Slusk4:     Slusk: = 0 ; N: = Nge ; Nido: = 0 ; for i: = 1 step 1 until N do ivar[i]: = ivarge[i] ; MINUT ;
            goto PROVUT1 end SLUSS ;
STEG:       begin integer Nvaks ; input (60, 'N', N) ;
            for i: = 1 step 1 until N do begin
                input (60,'2(N)',ik,w) ; ivar[i]: = ik ; if w > 0 then dark[ik]: = - w end ;
            if Orvar = 1 then MINUT ;
            if not Koks then goto KNUT ;
            input (60,'2(N)',Nskott,Nvaks) ;
            if Rurik = 19 then begin
              for Rs: = Rs1 step 1 until Rs2 do Niks[Rs]: = Nvaks ;
              for i: = 1 step 1 until Nvaks do begin input (60,'2(N)',ik,w) ;
                for Rs: = Rs1 step 1 until Rs2 do begin Vaks(Rs,i): = ik ; if w > 0 then darks
                    [Rs,ik]: = - w end end end ;
            if Rurik = 20 then begin
              for Rs: = Rs1 step 1 until Rs2 do Niks[Rs]: = 0 ;
              for j: = 1 step 1 until Nvaks do begin input (60, '3(N)',Rs,ik,w) ; i: = Niks[Rs]: = Niks
                    [Rs] + 1 ; Vaks[Rs,i]: = ik ; if w > 0 then darks[Rs,ik]: = - w end end ;
            for Rs: = Rs1 step 1 until Rs2 do if Niks[Rs] ⩾ Np[Rs] then Niks[Rs]: = 0;
            goto KNUT end STEG ;
STYRE:       = part 7
SÄRK:       begin
procedure   Koin ; begin  Tage: = false ; Orvar: = Orvarko ; Slusk: = Sluskko ; Prov: = Provko ;
                    Mink: = Minkko ; N: = Nko ; Nge: = Ngeko ; Nido: = Nidoko ; Ri: = Riko ;
                    Rj: = Rjko ; Rido: = Ridoko ; U: = Uko ; U1: = U1ko ; U2: = U2ko ;
                    Uc: = Ucko ; Umin: = Uminko ;
```

```
          for i: = 1 step 1 until Nido do ido[i]: = idoko[i] ;
          for i: = 1 step 1 until N do begin ivar[i]: = ivarko[i] ; kc[i]: = kcko[i] ;
            kmin[i]: = kminko[i] ; kv[i]: = kvko[i] ; pinne[i]: = pinneko[i] ;
            stek[i]: = stekko[i] end ;
          for i: = 1 step 1 until Nge do ivarge[i]: = ivargeko[i] ;
          for i: = 1 step 1 until N do for j: = 1 step 1 until N do begin s[i,j]: = sko[i,j];
            SH[i,j]: = SHko[i,j] ; ruta[i,j]: = rutako[i,j] end end Koin ;
procedure Kout; begin Tage: = true ; Orvarko: = Orvar ; Sluskko: = Slusk ; Provko: = Prov ;
                    Minkko: = Mink ; Prov: = false ; Slusk: = 0 ; Uko: = 0 ; Nko: = N ; Ngeko:
                    = Nge ; Nidoko: = Nido ; Riko: = Ri ; Rjko: = Rj ; Ridoko: = Rido ; U1ko: =
                    U1 ; U2ko: = U2 ; Ucko: = Uc ; Uminko: = Umin ;
          for i: = 1 step 1 until Nido do idoko[i]: = ido[i] ;
          for i: = 1 step 1 until N do begin ivarko[i]: = ivar[i] ;
            kcko[i]: = kc[i] ; kminko[i]: = kmin[i] ; kvko[i]: = kv[i] ;
            pinneko[i]: = pinne[i] ; stekko[i]: = stek[i] end ;
          for i: = 1 step 1 until Nge do ivargeko[i]: = ivarge[i]
          for i: = 1 step 1 until N do for j: = 1 step 1 until N do begin sko[i,j]: = s[i,j] ; SHko[i,j]:
                    = SH[i,j] ; rutako[i,j]: = ruta[i,j] end end Kout ;

          if not Tage then begin Kout ; Rs: = Rs1 - 1 ; goto Nysa end ;
          Rskott: = Rskott - 1 ;
          if Rskott > 0 then goto Särkut else Uko: = Uko + Umin ;
Nysa:     Rs: = Rs + 1 ; if Rs > Rs2 then goto Särkslut ; N: = Niks[Rs] ; SIKREN(N) ; Orvar: = 0 ;
          for i: = 1 step 1 until N do ivar[i]: = Vaks[Rs,i] ;
          if Skrikut > - 1 then SKOTT ; Rskott: = if N = 0 then 1 else Nskott ;
Särkut:   for i: = 1 step 1 until N do begin ik: = ivar[i] ; STEKA ; kv[i]: = kc[i]: = ks[Rs,ik] end ;
          goto LETA ;
Särkslut: Koin ; if Prov and Skrikut > - 2 then PROVAR ;
          if Skrikut > - 2 or Prov then UVAR ; if Skrikut > - 1 then STRECK ;
          goto UBBEUT end SÄRK ;
UBBEUT:   if Orvar = - 2 then Orvar: = - 1 ; if Orvar = 0 then Orvar: = 1 ;
          if N = 0 then begin Prov: = true ; MINUT end ;
          if N = 1 then goto ENSAM ; if Prov then goto PROVA ; goto LETA ;
PUTS:
UBBE:      = special parts 9, 10 etc.
FINAL:    end LETAGROP ;
```

SLUSS takes action if this is not so but the results indicate that at least one or two parameters are far off from the "best" values.

ENSAM treats cases with only one parameter to be adjusted and is designed to find the right minimum even with very bad initial guesses.

SÄRK governs the two-level variation of **k** and $\mathbf{k}_s$.

STYRE is a "species selector" which tries to compare various chemical models, especially to add new species in order to improve U.

The only parts of the program that are specific for a certain definition of U are PUTS (for pretreatment of data after the block DATA) and UBBE. By using switches in UBBE and PUTS, governed by the control number *Typ*, it is possible to combine the main program with special programs for several different problems in the same tape or disk.

We shall now discuss some features that are new compared with part 4.

Variation of parameters in LETA

Let us recall some notations. Each common parameter has one number, called ik (from 1 to N_k) among all $k[ik]$ and another number, i, among the N parameters to be

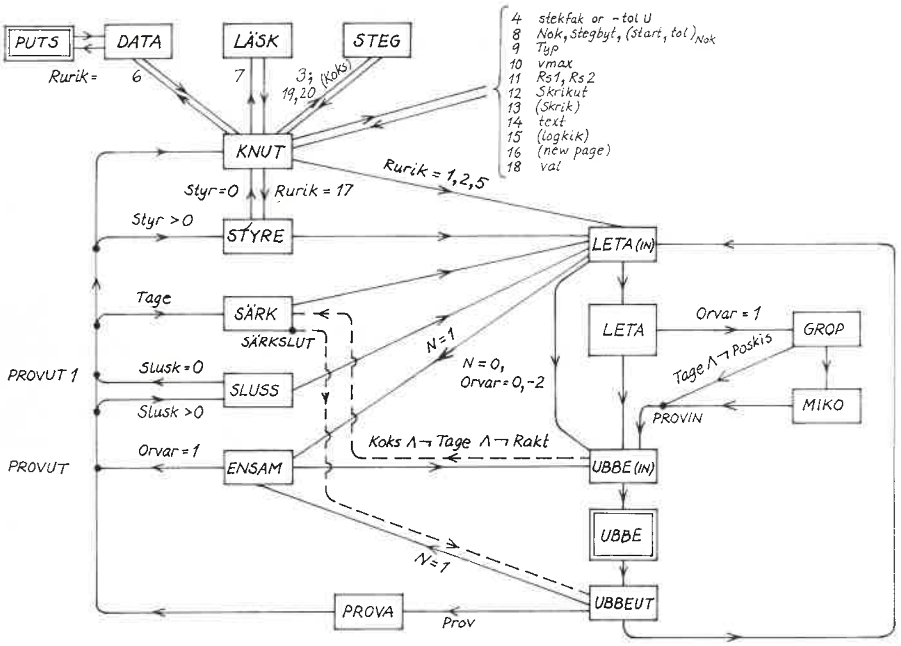

Fig. 1. Plan of present LETAGROP program, with the various blocks. Conditions for choosing one line or another are written close to some of the lines in the flowsheet. If a number stands alone, it means a value for *Rurik*. The program is given in Table 1. One transition PROVA→STYRE, and one MIKO→STYRE, have been left out for clarity.

adjusted. The relationship is expressed by a function *ivar:* if we are adjusting $k[2]$, $k[3]$, $k[4]$ and $k[6]$ out of $N_k = 6$ parameters, then $N = 4$, $2 = ivar[1]$, $3 = ivar[2]$, $4 = ivar[3]$ and $6 = ivar[4]$. In the program we write $k[ik]$ and $kv[i]$ so that in the example given, $k[6] = kv[4]$ etc; there are corresponding functions for the lower level variation of the $k_s[Rs, ik]$. In the discussion we shall mainly consider those parameters that are being adjusted, and name them by their i numbers.

Normal case: positive Orvar

In LETA, **k** is varied according to (1). When only one of the v_i is $\neq 0$, the corresponding i value is called *Ri* in the program; when two v_i are $\neq 0$, the program uses *Ri* and *Rj*. (In the program, i and j are used as "waste" integers, which may change when left out of sight.) The variation is regulated by the control number *Orvar*. The table below indicates, for a case with $N = 4$, the succesive values of *Ri* and *Rj*, the corresponding values for *Orvar* in UBBE and LETA, and the members of $\vec{\mathbf{p}}$ and **R** calculated in LETA.

Orvar is set $:= 0$ at the beginning of the calculations, or wherever an earlier minimum is not relevant: in KNUT on reading new data (Rur6), reading new parameters (Rur7), changing *Typ*(Rur9), changing groups to be treated (Rur11), changing definition of U(Rur18) and at appropriate places during the lower level variation in SÄRK.

If $Orvar = 1$ already when entering LETA, which means that the central point

Varied

Ri(*Rj*)	*Orvar* in UBBE	*Orvar* in LETA	Calculated in LETA
0	$0 \to 1$	$\to 2$	U_c
1+	2	$\to 3$	
1−	3	$\to (4)2$	$p_1 r_{11}$
2+	2	$\to 3$	
2−	3	$\to 4$	$p_2 r_{22}$
2(1)	4	$\to 2$	$r_{12} r_{21}$
3+	2	$\to 3$	
3−	3	$\to 4$	$p_3 r_{33}$
3(1)	4		$r_{13} r_{31}$
3(2)	4	$\to 2$	$r_{23} r_{32}$
4+	2	$\to 3$	
4−	3	$\to 4$	$p_4 r_{44}$
4(1)	4		$r_{14} r_{41}$
4(2)	4		$r_{24} r_{42}$
4(3)	4	$\to 1$	$r_{34} r_{43}$

$U_c(\vec{\mathbf{c}})$ coincides with an earlier found minimum $U_{\min}(\vec{\mathbf{k}}_{\min})$, then U_c is known and the program proceeds directly to *Orvar* = 2.

The present order of variation differs from that in part 4; it has some advantages for special programs, and for the adjustment of the steps, as follows:

As soon as each parameter *kv*[*i*] has been varied alone a test is made on the U values, U_1 (*kc*[*i*] + *stek*[*i*]) and U_2 (*kc*[*i*] − *stek*[*i*]), after the label Leta3. First, the quantity

$$\Delta = \tfrac{1}{2}(U_1 + U_2) - U_c \tag{7}$$

must not exceed $2U_c$/(number of data points) $\approx 2\sigma^2(y)$; $\sigma^2(y)$ is the squared standard deviation of the quantity y, the "error square sum" of which is used as the definition for U.

For comparison, for a shot around the real minimum, with steps = *stek*[*i*] = $F_{\text{stek}} \cdot \sigma(k)$, we would have $\Delta \approx F_{\text{stek}}{}^2 \sigma^2(y)$; with the standard value $F_{\text{stek}} = 0.5$, $\Delta \approx 0.25\, \sigma^2(y)$. Hence, if $\Delta > 2\sigma^2(y)$, *stek*[*i*] is decreased in the procedure Minskasteg, to avoid disturbance by higher-degree terms in $U(\vec{\mathbf{k}})$.

On the other hand, if neither $|U_1 - U_c|$ nor $|U_2 - U_c|$ exceeds $10 \times$ *tol* $U \times U_c$, there is a risk that the influence of *kv*[*i*] may be distorted by the noise of rounding, and *stek*[*i*] is then increased by the procedure Stegupp. The tolerance *tol* U begins at 10^{-6} and may be changed by *Rurik* = 4.

The lowest U value met with, $U_{\min}$, in set equal to U_c if *Orvar* = 0 or 1 on entering LETA (procedure MINUT). If a still lower value for U is later found during the variation, this is recorded by MINUT, together with the corresponding *kv* values (*kmin*). The corresponding values for *Ri* and *Rj*(*ido*[1] and *ido*[2]) are also noted; if *Rj* = 0, then *Nido* = 1 else *Nido* = 2. If, in our example above, the $U_{\min}$ was obtained in the set where *kv*[2] = *k*[3] and *kv*[4] = *k*[6] were both varied, then *ido*[1] = 2 and *ido*[2] = 4. The *ik* values may be expressed by *ivar*[*ido*[1 or 2]], for instance 6 = *ivar*[*ido* [2]].

All this information will be used if the calculated minimum point gives a value for U that is $> U_{\min}$. ("Slumpskott", see SLUSS below.)

Protected parameters; negative Orvar in LETA

For some parameters *k*[*ik*], such as equilibrium constants, a negative value has no physical meaning. In LETAGROP such parameters are "protected" against becoming

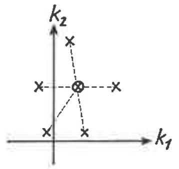

Fig. 2. Shows schematically how a protected parameter (k_1 or k_2) might become negative during the variation in LETA. This risk is avoided by procedure PLUSKA.

negative by setting the corresponding Boolean *posk*[*ik*] **true**, after which only positive values or zero will be accepted in the calculations. For non-protected parameters *posk*[*ik*] is **false**.

As the computer is about to start a new shot (be it the first one, or not) the values for $\vec{\mathbf{c}}$, **S** and **H** may be such that a negative *kv*[*i*] might appear in some set during the variation by (1) (see Fig. 2). If *kv*[*i*] is a "protected" parameter, the value would be meaningless and, worse, the calculated U would really not be on the surface $U(\vec{\mathbf{k}})$ since in most special blocks UBBE, a negative equilibrium constant would be treated as if it were zero.

To avoid these risks, a test is made in procedure PLUSKA, which if necessary increases the central values *kc*[*i*] for the threatened parameters.

If the *kc*[*i*] corresponded to a minimum before PLUSKA (all *kc*[*i*]$=$*kmin*[*i*]), then *Orvar* was at first 1 ("we have $U_c = U_{\min}$"). After the adjustment in PLUSKA, some *kc*[*i*]$\neq$*kmin*[*i*]. Orvar is then set $:= -2$ ("we have $U_{\min}$ but no U_c"). In UBBE, U is calculated for the new $\vec{\mathbf{c}}$, and Orvar set $:= -1$ ("U_c calculated but not tested"). It is tested in procedure Komner. If $U - U_{\min} \leqslant 2\ \sigma^2(y)$ (or $\leqslant 0.04\ U_{\min}$, if $\sigma(y)$ is not available), the new $\vec{\mathbf{c}}$ is accepted, and the LETA variation is continued (*Orvar* $:= 2$). If $U - U_{\min}$ is greater, however, the central point is considered to have been taken too high up the walls of the pit; "KOMNER" ($=$come down) is printed, and either some term in **H**(*stek*[*i*]) or some in **S**(*s*[*i*, *j*]) is decreased. Then all *kc*[*i*] are at first taken back to the original minimum, and PLUSKA is again applied, *Orvar*$:= -2$, U is calculated etc.

When some stek value is increased by Stegupp, a test must also be made in PLUSKA that no protected parameter can turn negative. If this necessitates a change in some *kc*[*i*], *Orvar* is set$:= -5$ and U calculated for the new $\vec{\mathbf{c}}$ and tested in Komner. If it is accepted, *Orvar*$:= -4$ and U_1 is calculated (with *kv*[*i*]$:=$*kc*[*i*]$+$*stek*[*i*]), then *Orvar*$:= -3$, and U_2 is calculated (with *kv*[*i*]$:=$*kc*[*i*]$-$*stek*[*i*]). If the values for U_1, U_2 and U_c again lead to a change of *stek*[*i*] by Stegupp, and of some *kc*[*i*] by PLUSKA, then *Orvar*$:= -5$, and the cycle is repeated. Otherwise U_c is accepted, *Orvar*$:=2$, *Ri*$:=1$ and the shot is started all over again.

Adjustment of a single parameter in ENSAM

The block ENSAM is designed to adjust a single parameter to give the minimum value for U, even in the case of very bad initial guesses.

The need arises e.g. in equilibrium analysis when one starts from a model with a certain set of species, has adjusted their formation constants to fit the data, and wishes to add a new species and see if this improves the agreement. It may then be

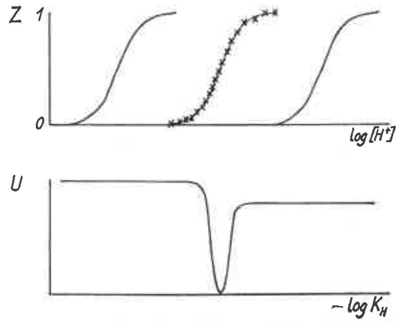

Fig. 3. (Schematical) a) Measurements of protonation of anion base, Z as a function of $\log h$. b) U as a function of the value assumed for the protonation constant, $\log K_H$: note deep minimum and two high plateaus.

hard to guess a good value for the formation constant of the new species *a priori*. If one wants to test, say, 10–20 possible new species one would like to be able to start with very crude guesses, which may be many powers of ten off the right order of magnitude, and let the computer find the right range.

A very bad guess for a parameter, say k_1, means that one has landed at a point where the curve $U(k_1)$ is almost horizontal. For instance, the schematic fig. 3 shows data for the protonation of an ion A^- with the protonation constant $K_H(H^+ + A^- \rightleftharpoons HA)$. The ordinate, Z, is the number of protons bound per A ($Z = [HA]/([A^-]+[HA])$. If K_H is chosen much too small then the calculated value is $Z_{calc} \approx 0$ in the whole range, $U = \Sigma(Z_{calc} - Z_{exp})^2 \approx \Sigma Z_{exp}{}^2$. On the other hand, if the value guessed for K_H is much too high, $Z_{calc} \approx 1$, and $U \approx \Sigma(1 - Z_{exp})^2$. If we plot U as a function of the log K_H assumed, there will be a deep minimum at the correct value for $\log K_H$, and on each side of it a flat plateau extending to positive or negative infinity. (With $U(K_H)$ of course the curve ends at $K_H = 0$.)

The general strategy of LETAGROP for adjusting a single parameter, say k_1, is to calculate three points on the curve $U(k_1)$ namely $U_c(kc_1)$, $U_1(kc_1 + stek_1)$ and $U_2(kc_1 - stek_1)$, to calculate the position k_{01} of the minimum of the parabola through these points, and to test whether $U(k_{01})$ is an improvement or not.

If one has started on a plateau such as that described in fig. 3, at least one of two awkward things will happen: (1) The variations because of rounding errors in the computer are larger than the real bend in the surface, so that the calculated minimum is spurious. (2) The curve $U(k_1)$ does bend in the range studied, but the bend is convex upwards, so that the calculated k_{01} gives the maximum in the convex parabola through the three points, which is in the direction opposite to the minimum one is searching for.

Hence, before it tries to calculate the position of the minimum, the computer should first find a range where the curve $U(k_1)$ is certainly concave.

ENSAM replaces LETA, GROP, PROVA and MIKO whenever $N = 1$. Its strategy is slightly different for protected and non-protected parameters.

ENSAM for non-protected parameters

In ENSAM, too, the control number is called *Orvar*. The computer starts with the initial values for the parameter, *kc*, and the step, *stek*, and calculates the three U values:

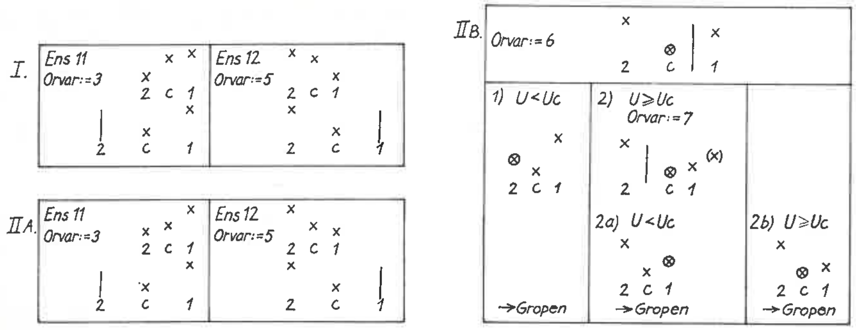

Fig. 4. Some possible situations during the variation of a single non-protected parameter in ENSAM. See text.

$U_c(kc)$, $U_1(kc+stek)$ and $U_2(kc-stek)$, with $Orvar=1$, 2 and 3. Provided that the parabola through these three points is significantly concave, the coordinates of its minimum (*Uno*, *kbom*) are calculated after the label Gropen. The calculated *kbom* is tested ($Orvar=4$). If it really gives a lower U than earlier known points, this is accepted as the new minimum at Träffen (procedure MINUT). Otherwise the search is continued.

The behavior in some possible cases is indicated in Fig. 4, where U is the vertical axis and k_1 (or *kv*) is the horizontal axis. Crosses are calculated points, 2, c and 1 indicate the points with U_2, U_c and U_1, and a vertical line indicates the *kv* value for which U is next calculated.

I. If the curve through the three calculated points is convex or at least not significantly concave (Δ (eqn 7) $<10\times tolU \times U_c$), then *stek* is doubled and a new point calculated in the direction where a downward slope was found, however small. (If this was spurious, some following step will go in the other direction.) So the computer goes on until a concave triplet is found.

II. If there was a concave triplet, but the calculated minimum of the parabola through the points is not acceptable (U value too high) the search goes on. If U_c was not the lowest point, the program searches in the direction of the slope (Fig. 4 II A). If U_c was the lowest point, *stek* is halved and Fig. 5, II B shows the cases that may arise. In very rare cases (with a skew pit) the triplet may be narrowed as in II B 2b until the difference $\Delta < 10.5\times tol\, U \times U_c$, and hence begins to approach the rounding errors in the computer. In such cases U_c is accepted as the minimum, to avoid misbehavior. It is certainly not a bad approximation, for most purposes.

ENSAM for protected parameter

If *kv*[1] is "protected", ENSAM avoids its becoming negative in two ways:
(1) If the calculated value for *kv* at the minimum, *kbom*, is negative, then the program sets $kv:=0$, changes the Boolean *Mink* from **false** to **true**, sets $Orvar:=4$, calculates U and tests it at Ens4. If the value $kv=0$ gives an acceptable minimum, one more attempt is made (with $Orvar=10$, and then 5), using the U values for $kv=0$, *stek* and $2\times stek$; *stek* is calculated from the standard deviation *dark*. If a negative *kbom* is again found at Gropen, the value for U at $kv=0$ is known so the test is made at once

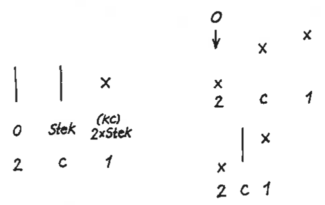

Fig. 5. Variation of protected parameter k in ENSAM in vicinity of $k=0$. See text.

at Ens4; if $U(0)$ is still the lowest value, $kv=0$ is accepted at Träffen as the result of the shot.

(2) In the variation after Ens11, kv might become negative. This is prevented as follows (Fig. 5 left). If $kc<3\times stek$, the program passes to Ens13, and U is calculated for $kv=0$ ($Orvar=8$) and the midpoint ($Orvar=9$).

If the triplet of points so found is concave, *kbom* is calculated at Gropen. If the calculation at Gropen gives a negative *kbom* while $Orvar=9$, it is tested whether the known U for $kv=0$ is acceptable as $U_{\min}$; if so, this is taken as the result of the shot.

If the triplet is convex, U_2 is kept at $kv=0$, and the interval is halved (Fig. 5 right) at Ens14. This is repeated until either a concave triplet is obtained (in which case *kbom* is calculated at Gropen) or it seems likely that the curve $U(kv)$ is convex down to zero. ($U_1<U_2+10\times tol\ U\times U_c$ and still not concave). In the latter case, the program passes to label Kasso, prints "KONVEX TILL NOLL" (=convex to zero), sets $kv{:}=0$, calculates $stek=\sigma^2(y)\ (\Delta k/\Delta U)_{k=0}$) and sets its negative value as *dark*, to give a reasonable step in the following variations with several parameters.

It may happen that the calculated $U_0<0$ ("minusgrop"), especially if the concave triplet had a wide span. If the test at Ens4 gives an acceptable new $U_{\min}$, the program sets, $stek{:}=0.5\times \text{abs}\ (dark)$, after which a new attempt is made with $Orvar=10$ and 5.

The present version of ENSAM has been working for a couple of years without any serious difficulties. One day Nature may provide us with a new type of misbehavior so that an extra safeguard is needed. Of course, if one tries to adjust a parameter that does not influence U (such as the formation constant for a species, one component of which has the concentration zero), then the computer will try to increase that parameter until its numerical value becomes too big for the computer. Also, if one starts with a guess that is off by several powers of ten, the parameter may at present approach to the "best" value only by the factor 2 or 0.5, hence by adding ± 0.3 to the decadic logarithm. This is not too serious if the error is only by a factor of 10^3 or 10^4, but if one plans to make a large number of guesses that may be too large or too small by a factor of 10^{20} or more, one might as well rewrite ENSAM so that it increases (or decreases) k faster.

Adjustment of several parameters: function of PROVA and SLUSS

If $N>1$, the parameters are varied by LETA, and the minimum $\vec{k}_0$(*kbom*[*i*] in the program) calculated by GROP. As described above, LETA (by procedure MINUT) records $U_{\min}$, the lowest U value met with, and the corresponding values for the adjusted parameters, *kmin*[*i*].

From GROP the program proceeds via MIKO to PROVIN. (If some protected parameter is negative at the calculated minimum, at this stage a reduced minimum is calculated in MIKO, as will be described in part 7.) Then UBBE is used to calculate the U value at the calculated minimum.

PROVA compares this U value with the earlier $U_{\min}$. Three possibilities arise:

(1) The new U value is lower than the earlier $U_{\min}$. It is then accepted as the new $U_{\min}$, by procedure MINUT after the label Träff. The calculated set of parameters $\vec{\mathbf{k}}_0$(*kbom*) is made to be the new central set $\vec{\mathbf{c}}$ (*kc*).

(2) The earlier central value U_c was lower than the U calculated for $\vec{\mathbf{k}}_0$. In this case the computer prints "GAMLA KONSTANTER" (=old parameters) and keeps the *kc* set. This behavior is met with especially under two sets of circumstances:

(2*a*) The calculated minimum often comes close to the real minimum after one or two shots. In following shots, because of small rounding errors in the computer, the calculated minimum will begin to jump around in a small range around the real minimum. Sometimes a new shot gives a lower value for U, sometimes a slightly higher one, but the change in the parameters is always only a small fraction of their standard deviations.

At present, our general policy is to allow two, three or four shots and to be satisfied when $U_{\min}$ stays practically constant. In a future edition one might leave the judgment to the computer and let it make new shots until either U at the calculated minimum differs only by a certain fraction, say *tol U* from earlier $U_{\min}$, or until "GAMLA KONSTANTER" appears.

(2*b*) In a few cases we first found a vector $\vec{\mathbf{k}}_0'$ that gave a certain low value U', but the next shot, and all the following, converged on a vector $\vec{\mathbf{k}}_0''$ which was not far from the $\vec{\mathbf{k}}_0'$, and which gave a slightly higher value U''; moreover some values for the standard deviations $\sigma(k_i)$ could not be calculated since the terms under the square root (eqns 3:27, 28) were consistently negative. The computer in this case prints, instead of the σ's, a negative number, $-|stek|$. Since $U(\vec{\mathbf{k}}_0')$ was the lowest value met, each parameter was varied around $\vec{\mathbf{k}}_0'$ and the steps were as usual $\pm F_{\text{stek}} \times \sigma(k_i)$ or, to be strict, $\pm F_{\text{stek}} \times h_i \times \sigma(v_i)$, compare Fig. 2, part 3).

Usually, F_{stek} (*stekfak* in the program) is set equal to 0.5. The failure to calculate some $\sigma(k)$ indicates that the calculated second-degree surface is not an elliptic paraboloid with a minimum, but a hyperbolic paraboloid with a saddle point. Since the real surface must have a minimum (after all, a sum of squares cannot turn negative), we conclude that the deviations because of higher-degree terms are important even at a distance of about 0.5 $\sigma(k)$ from the minimum.

Indeed, in the cases mentioned, it was found that if F_{stek} was instead set equal to 0.2 or 0.1, the calculated minima converged quickly to a quite normal minimum with all $\sigma(k)$ positive. If these cases were to be more frequent (we have met them only twice when two species with very similar composition were assumed to exist simultaneously), one might let the program decrease F_{stek} automatically.

(3) U at the calculated minimum is higher than $U_{\min}$, and $U_{\min}$ was met with on varying one or two parameters during LETA. In this case the computer prints "Slumpskott" (=hit by chance). In the edition of part 4, the set $\mathbf{k}$ corresponding to $U_{\min}$ was simply used as the new central set for the parameters, which took one closer to the real minimum, step by step. Some experience gave us the idea for a more efficient treatment, however.

In the present edition, a Slumpskott automatically starts a series of calculations, which speed up the convergence. The governing block for these calculations is SLUSS which uses a control number *Slusk*. A Slumpskott is an indication that some parameters are far off from their values at the real minimum, and the most suspect ones are those that were varied when the $U_{\min}$ was met with, and which were recorded by LETA. Hence, if one parameter was varied ($Nido = 1$), this parameter is adjusted by means of ENSAM, keeping all others constant. If $U_{\min}$ was met when two parameters were varied ($Nido = 2$), then ENSAM adjusts first one of these alone, then the other one alone, and finally both are adjusted together, in two shots using LETA. If one of the latter two-variable shots would give a second "Slumpskott", this is noted but cannot start a similar series as long as SLUSS is in action ($Slusk > 0$).

Strategy for fast convergence

Thanks to the mechanisms described, in ENSAM, PROVA and SLUSS, there is in our experience no risk that the calculations will diverge, since U will always decrease, or at the worst keep constant. In the very worst case described under (2*b*) ("Gamla konstanter") the calculations may converge to a point at a short distance from the best minimum, and this case may be recognized and corrected for by decreasing the F_{stek}.

At any rate, for aesthetic and economic reasons, one wishes to reach the minimum in the shortest possible time, and the quickest way can only be approached by a combination of judgment and experience. If one starts with, say, seven or eight possible parameters, we have found it advantageous first to adjust the three or four most important ones in one or two shots, and then to add the others. If one knows that the guessed values for one or two parameters may be extremely bad, it may be economical to keep the others constant and vary the suspect ones alone before one proceeds. After all, to calculate a minimum for two parameters one needs only to calculate six values for U, whereas for $N = 7$ one has to calculate $\frac{1}{2}(7+1)\ (7+2) = 36$ and for $N = 8$, 45 U values. In any case the machine will finally find a minimum.

Another possible way is just to start with one or two parameters and find some sort of a minimum, however bad, and after that to add one new parameter after another. Such a strategy is described in detail in part 7 which deals with the "species selector".

Two-level adjustment of k and k_s: function of SÄRK

The "third strategy" for adjusting common and group parameters is governed by the block SÄRK. In the input (see below) one tells which k_s values are to be adjusted. If one wishes to adjust exactly the same k_s parameters in all groups, one may use $Rurik = 19$. With $Rurik = 20$, on the other hand, one may "hand-pick" the k_s parameters to be adjusted in each group.

Since the same equations are to be used for the adjustment both at the upper level (adjusting the k) and at the lower level (adjusting the k_s), it is convenient to use the same blocks, LETA etc. While the computer is working at the lower level, quantities necessary for adjusting the k parameters at the higher level are "saved" by the procedure Kout as quantities with names ending with "$-ko$" (*Riko*, *Sluskko*, *SHko* etc); they are restored by procedure Koin on returning to the higher level. The number of shots allowed to adjust the k_s for each group at the lower level is given (as N_{skott}) together with $Rurik = 19$ or 20. The sum of the minimum contributions to U, "*Uko*" is used on the upper level as the U value for that set of k values.

The upper level works just as if k parameters are varied alone. On the lower (*Tage*) level the blocks also work in much the same way: ENSAM for a single parameter, LETA + GROP + PROVA for several, SLUSS in the case of "Slumpskott" etc. The only difference is that MIKO is called for only in special cases (one of which happens to be the spectrophotometric case), and even then the procedures PLUSKA and Komner are not needed so that there will be no negative *Orvar* on the lower level.

In the present program, the values for the k_s that correspond to a minimum on the upper level are also stored as *ksmin*. The calculated standard deviations for the k_s are stored as *darks* and *darks2*. A twist matrix is used for the k_s starting from the second shot on the lower level. However it does not seem worth the extra space to save all the twist matrix terms for the group parameters, as is done for the common parameters.

Input

While developing a program which is being continuously used by this and other laboratories we have tried to avoid as far as possible to make old input cards unacceptable; this has among other things led to a somewhat random order of the Rurik numbers.

As in part 4, the input is conveniently listed after the corresponding Rurik numbers. Comments are given here only where there is a difference from the version in part 4. In the input lists, a subscript gives the number of repetitions.

Rurik = 1. One U calculated with available parameters, as in part 4.

Rurik = 2. "Uttåg", data of interest printed for each point, as in part 4.

Rurik = 3, N, $(ivar[i], w)_N$. Preparation for shot, varying only the N common parameters $k[ik]$ with the numbers $ik = ivar[i]$, $i = 1 \rightarrow N$. If w is positive, it is stored as $dark[ik] = -w$, if w is negative, it is disregarded. After *Rurik* = 5, procedure STEKA translates this information to the step *stek*[i] for variation. If *dark*[ik] is negative, either it has come from the w given after *Rurik* = 3 or from the "-1" given in LÄSK, then *stek*[i]: = |*dark*[ik]|. If *dark*[ik] is positive, it is equal to the calculated standard deviation $h_i\sigma(v_i)$ and then *stek*[i]: = *stekfak* × *dark*[ik]. Hence, if *Rurik* = 3 is given several times, with variation of the same parameter, a positive w (first step) should usually be given only the first time.

Rurik = 4, stekfak. If a value for the standard deviation (in twisted space) *dark*[ik] or *darks*[Rs, ik] has been calculated, and the quantity is positive, then the step *stek*[i] in the next variation is the product of *dark* or *darks* with *stekfak*. *Stekfak* is automatically set : = 0.5 in the beginning of the calculations, but if the pit deviates strongly from a pure second degree function, it may be advisable to set a lower value such as 0.2 or 0.1 by giving in the input e.g. "4, 0.1,". The new *stekfak* then remains until it is again changed by *Rurik* = 4.

Rurik = 4, $-$*tolU*. The tolerance *tolU* (see ENSAM above) is set: = 10^{-6} by the program but may be increased if the U values scatter because of rounding errors. The scatter may also be decreased by decreasing the tolerances after *Rurik* = 8.

Rurik = 5. A shot is made, with the instructions given earlier, after *Rurik* = 3, 19 or 20.

Rurik = 6, Ns, Nag, Nas, Nap, $(ag)_{Nag}$, $(Np, (as)_{Nas}, ((ap)_{Nap})_{Np})_{Ns}$. The (invariable) data, *ag*, *as*, *ap*, are given in the order described in part 4. Examples for special problems are given in following parts. The matrix *ap*, which is the largest matrix in the program, now stores the data in a more economical way than earlier so that the program is now very flexible as to the number of points per group ($Np[Rs]$) and the number of data per point (Nap = read in input, $Napa$ = total number of cells reserved per point, including those reserved for values to be calculated later).

Rurik = *7* (common), (group), (twist). Information on adjustable parameters. Alternatives for the three parts:

(common) = (a) Nk, Nk, Nak, $(k, (ak)_{Nak})_{Nk}$; new problem. (b) Nk, $Nbyk$, $(ik, k, (ak)_{Nak})_{Nbyk}$; partial change, or addition of $Nbyk$ k values. (c) Nk, 0; no change

(group) = (a) Nks, Nks, $((ks)_{Nks})_{Ns}$; new problem, same if $Nks = 1$ or 0. If $Nks = 0$, only 0,0 is given. (b) Nks, 1, j, $(ks)_{Ns}$; $ks[Rs, j]$ is exchanged in all sets. (c) Nks, -1, m, $(j)_m$; $ks[Rs, j]$ are set $:= 0$ for all the j values given. (d) Nks, 0, m, $(j)_m$, $((ks[Rs, j])_m)_{Ns}$; all ks: $= 0$ except m of them in each group. (e) Nks, -2; no change.

(twist) = (a) $skin$, $(ik, jk, sk)_{skin}$; values given for some twist matrix terms. (b) 0; no change.

Rurik = *7*, Nk, -1, $Negk$, $(ik)_{Negk}$. The *posk* protection is removed from $Negk$ of the k parameters, which are thus allowed to be negative.

Rurik = *8*, Nok, *stegbyt*, $(start, tol)_{Nok}$. This is information for solving equations in UBBE and will be discussed in part 8.

Rurik = *9*, *Typ*. The number *Typ* tells the type of problem and picks out the right label in various switches in PUTS and UBBE.

Rurik = *10*, *vmax*. Gives the number of loops in procedure Kålle after which x and y values are printed, as a diagnosis that the input after *Rurik* = 8, (see part 8) should be changed. In the beginning of the program, v_{max} is automatically set $:= 120$.

Rurik = *11*, *Rs1*, *Rs2*. Serial numbers of first and last group to be treated, as in part 4.

Rurik = *12*, *Skrikut*. *Skrikut* is a control number, which may at present have the values 1, 0, -1 and -2. It is used to suppress certain types of output. For instance, in a two-level adjustment (*Koks* = **true**) one is seldom interested in all details of the lower-level shots, and when the species selector (part 7) is used, one is often satisfied with the final result for each new species attempted. The following information is printed:

I. In variation of k (**not** Tage) (*a*) *kv* and U values during a shot, Minskasteg, Stegupp, Komner, s terms: always for *Skrikut* = 1 or 0; if *Koks* is **true**, also for *Skrikut* = -1, otherwise not. (*b*) Minusgrop, *kbom*, *darr*, *sigy* ($\sigma(y)$), U at test in PROVA, Gamla konstanter, Slumpskott, *Umin* and *kmin* for Slumpskott, partial results of MIKO: for all *Skrikut* values.

II. In variation of k_s (*Koks* **and** *Tage*), (*a*) *kv* and U during a shot: only for *Skrikut* = 1, (*b*) Minska steg, Stegupp, *kbom*, *darr*, *sigy*, s terms, U at PROVA, Gamla konstanter, partial results of MIKO: only for *Skrikut* = 1 or 0, (*c*) Slumpskott, *Umin* and *kmin* for Slumpskott: for *Skrikut* = 1, 0 or -1 but not for -2, (*d*) Minusgrop: for all *Skrikut* values.

Rurik = *13*. Available "best" values for parameters k and k_s are printed by procedure SKRIK, with their calculated standard deviations (*darr*), as in part 4.

Rurik = *14* / text on next card. Text on following card is printed. Can of course be repeated: 14 / text / 14 / text ...

Rurik = *15*. The information on the $k[ik]$ and their standard deviations is transformed to decadic logarithms. (Procedure Logkik). The limits given correspond approximately to $\log(k \pm 3\sigma(k))$; if $\sigma(k) > 0.2 \times k$, only the "best" value for $\log k$, and the maximum value, $\log(k + 3\sigma(k))$ are given. If no standard deviation is available, or if $k + 3\sigma(k) < 0$, this is indicated.

Rurik = *16*. Gives new page in output.

Rurik = *17* + information. This starts one cycle of the species selector (part 7).

Rurik = *18, val*. The number *val* tells which deviation *fel*[*val*] is to be used in calculating U. *Val* is automatically set : = 1 after Rurik = 9 (new *Typ*).

Rurik = *19*, N, $(ik, w)_N$, *Nskott*, *Nvaks*, $(ik, w)_{Nvaks}$. Preparation for shot with *Koks* **true**. N of the common parameters are to be varied (information as for Rurik = 3) on an upper level. For each set of k parameters during a shot on the upper level, N_{skott} shots are to be made for adjusting the same set of k_s in each group. The serial numbers of the latter are given, together with w: w is the stek if positive, and if w is negative, *stekfak* × |*darks* [*Rs, ik*]| is used for *stek*. For adjusting only k_s parameters and no common k (earlier *Rurik* = 10, part 4), one may use *Rurik* = 19 with $N = 0$.

Rurik = *20*, N, $(ik, w)_N$, *Nskott*, *Nvaks*, $(Rs, ik, w)_{Nvaks}$. As for *Rurik* = 19 except that the *ks*[*Rs, ik*] parameters to be adjusted are handpicked. For example, in a case with $N_s = 4$, $N_{ks} = 4$ we might wish to adjust the k_s values marked by a cross below:

Group no.	1				2				3				4			
Group parameters k_s no.	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Rurik = 19:		×		×		×		×		×		×		×		×
Rurik = 20:		×		×					×				×	×	×	×

In the two cases, the input would be:

19, N, $(ik, w)_N$, N_{skott}, 2, 2, w, 4, w,

20, N, $(ik, w)_N$, N_{skott}, 7, 1,2, w, 1,4, w, 3,1, w, 4,1, w, 4,2, w, 4,3, w, 4,4, w,

Order of input

The type of information can be characterized simply by the *Rurik* numbers. The following starting order is advisable; other *Rurik* numbers may be inserted and added at will:

Fixed input: 14 (headline for output), 9 (*Typ*), 6 (data).

Variable input ("dagens spaning"): 7 (parameters), 8 (information on equation solving), 11 (first and last groups), 18 (*val*, if ≠ 1), 2 (if initial Uttåg is desired); 3 (or 19 or 20, steps), 5, 5, (two shots) ...

Note that *Rurik* = 2 automatically sets N: = 0 and hence must not be placed immediately after *Rurik* = 3, 19 or 20. It may be used between shots when *Koks* is **true**, e.g. 5, 5, 2, 5, 2.

The input may end with 13 (k and k_s printed with σ values), 15 (log k printed), 2 ("Uttåg"), −1 (finis);

"Multiple minima"?

It is sometimes asked what our programs do with "multiple minima" or "local minima", the idea being that there may be several minima in the function $U(\mathbf{k})$. The answer is that in the various types of problems that we have treated we have so far not found a single case with more than one minimum that could be said to correspond to a fit. Whether or not there are additional shallow minima on high plateaus in the U function (compare Fig. 3) is a problem that has not interested us.

In this connection we may point out three types of misbehavior that may easily be recognized and corrected for by changing the input.

(1) The calculation of the errors often involves the solving of one or more equations, for instance mass balance equations. If the tolerance for solving these equations is set too high, there will be a corresponding random noise added to the U values, which may even give the idea of a cluster of local minima. If the tolerance limits after *Rurik* = 8 are lowered sufficiently, these symptoms disappear and a single minimum is found.

(2) By negligence one may try to adjust two interdependent parameters independently. For instance, one may happen to give the formation constant for the same species twice in the list of parameters, (say, as k_2 and k_8), All combinations $\vec{\mathbf{k}}$ with the same value for $(k_2 + k_8)$ will then be equivalent and give the same U value, and instead of a minimum point there is a ditch with a level bottom, so that LETAGROP will run along the bottom of the ditch without finding a satisfactory minimum. Again, this case is not hard to recognize and correct.

(3) A few data points may contain gross errors, either from the original data or, more often, from misprints in punching. These points may easily give a contribution to U which outweighs that of all other points. The whole calculation is then distorted by the attempts to diminish the errors stemming from these points.

Exactly for this reason, whenever new data are to be treated with LETAGROP it is good practice to ask for an "Uttåg" (*Rurik* = 2) in the very beginning of the calculations. Even with quite bad initial parameters the grossly erroneous points stand out, so that misprints may be corrected and gross experimental errors eliminated before one starts the main calculation

If the experimental data have an extremely large spread, so that no scientist would try to fit them to a model by graphical methods we might imagine that one might find cases with several shallow minima in the $U(\mathbf{k})$. Such data, however, would not be very interesting.

To sum up, if anyone has found a problem that could be treated by LETAGROP and that seems to lead to multiple minima, we would be glad to learn about it.

ACKNOWLEDGEMENTS

We wish to thank Professor Nils Ingri and Drs Robert Arnek, Hay Diet Liem and Olle Wahlberg for valuable discussions and a pleasant cooperation around various computers, and Mr Ashley Stevens, B. A. (Oxon) for revising the English of the manuscript. This work is part of investigations supported by Statens naturvetenskapliga forskningsråd (Swedish Natural Science Research Council). Computer time has been made available from 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).
ISO draft proposal for input, output, procedures for Algol 60 TSO/TC 97/SC 5 (Sec-24) (1965).
References to chemical applications will be given in following papers.

Tryckt den 26 augusti 1969

Uppsala 1969. Almqvist & Wiksells Boktryckeri AB