ARKIV FÖR KEMI Band 23 nr 10

4.64051 Read 11 March 1964

High-speed computers as a supplement to graphical methods

IV. An ALGOL version of LETAGROP VRID

By Nils Ingri and Lars Gunnar Sillén

CONTENTS

Introduction

In connection with our work on equilibria with polynuclear complexes, and on the thermochemistry of complex formation, we have developed a series of computer programs, using the general "pit-mapping" principle of LETAGROP [1, 2]. They

were written in the Ferranti autocode, an early version also in Fortran. In 1962, we introduced in our programs the principle of the twist matrix for treating skew pits [3]. These programs, called LETAGROP VRID (from Swedish vrida = turn, twist), were first also written in the Ferranti autocode. Later on we have worked out improved versions in the international computer language ALGOL.

Since these programs have proved quite useful to those chemists who have learned how to handle them, we now publish a relatively full description of the new LETAGROP VRID, in ALGOL. It takes a considerable effort to master all details in a program of this size. In the present paper a survey is given of the general plan, and enough detail is also given (we hope) to permit the reader to transfer the program to his own computer and problem and to check, and rewrite any separate part of it. For the mathematical background, the reader is referred to parts I–III [1–3].

The program tries to find the set of values for certain unknown parameters ("constants"), k and ks that will minimize a function U of the k and ks, and the data.

In the program given in Table 1 the special parts are designed for a very common type of data obtained in studying polynuclear complexes, namely $Z(\log a, B)$. The equations are:

$$B = b + \sum q\beta_{pq}\, a^p b^q \tag{1}$$

$$BZ = \sum p\beta_{pq}\, a^p b^q \tag{2}$$

$$Z_{\text{calc}} = Z + \delta Z \tag{3}$$

$$U = \sum w(Z_{\text{calc}} - Z_{\text{exp}})^2; \quad w = 1 \tag{4}$$

The β_{pq} in (1) are the $k[i]$ in the basic program, which are common to all experimental points. The δZ in (3) is a systematic experimental error which may be different for different groups of experiments; it corresponds to $ks[Rs,1]$ (Rs = number of group) in the basic program.

The calculations are based on the idea that U, in the neighborhood of the minimum, is approximately a second-degree function of the k and ks.

The general plan of the program is best shown by the diagram (Fig. 1). It consists of a number of blocks, DATA, LÄSK etc.; one of them, KNUT, is a switchboard to which the computer returns after fulfilling each task. Depending on which value for the control number *Rurik* it then reads, the computer jumps from KNUT to another block for a new task.

The *special blocks* PUTS and UBBE contain those equations that are special for the problem in question. All the other blocks—thus by far the greater part of the program—will be referred to as *basic blocks*. If one wants to change over to a completely new problem, for instance to treat data on ion exchange equilibria or extraction equilibria, or thermochemical data, then all the basic blocks can be used, thus all parts of the program with the exception of PUTS and UBBE which contain instructions on how to calculate U from the given data and constants.

Several sets of special blocks may be used in the same program; the right set can then be selected by means of a switch, controlled by the number *Typ*.

In the program given here, the special parts define U as an error-square sum. However, this is not necessary; the general parts can be used also to minimize other types of functions $U(k, ks)$.

Table 1. LETAGROP with VRID and MIKO, Spring 1964.

```
begin real det, ln10, sigy, sig2y, start, stegbyt, tol,U,U1,U2,Uc,Ucspar,Umin,Uno,w;
integer Rurik, Typ, Orvar, i,ik,ip,j,jk,m,N,Nag,Nak,Nap,Nas,Nge,Nimi,Nk,Nks,Ns,Ri,
    Rj,Rp,Rs,Rs1,Rs2,Rv; boolean Tage;
real array ag[1:10], ak[1:20,1:3], ap[1:25,1:40,1:6], as[1:25,1:6], dark, dark2[1:20],
    darks[1:25,1:5], darr1, darr2, dia1, dia2[1:12], k[1:20], kbom, kc, kcspar, kmin[1:12],
    ks[1:25,1:5], kv, pina, pinne[1:12], rucka, ruta, rut1, rut3, rutinv, s, SH, SHinv
    [1:12,1:12], sk[1:20,1:20], stek, stekge, v, vbom[1:12], vikt[1:25,1:40], vri[1:12,1:12];
integer array imi, ivar, ivarge[1:12], Np[1:25];
boolean array posk[1:20];
switch PUTS: = Puts1,Puts2;
switch UBBE: = Ubex1, Ubex2;
switch Letax: = Leta1, Leta2, Leta3, Leta4;

procedure Invert(N,A,eps,det, sing); comment matrixinversion by Gauss-Jordan elim-
    ination;
value N,eps; real eps,det; integer N; array A; label SING;
begin real y,w; integer i,j,k,L,p; array B,C[1:N]; integer array Z[1:N];
det: = 1;
for j: = 1 step 1 until N do Z[j]: = j;
for i: = 1 step 1 until N do begin k: = i; y: = A[i,i]; L: = i - 1; p: = i + 1;
for j: = p step 1 until N do begin w: = A[i,j];
if abs(w) > abs(y) then begin k: = j; y: = w end end;
det: = y×det;
if k ≠ i then det = - det;
if abs(y) ≤ eps then goto SING;
for j: = 1 step 1 until N do begin C[j]: = A[j,k]; A[j,k]: = A[j,i]; A[j,i]: = - C[j]/y; B[j]: =
    A[i,j]: = A[i,j]/y end;
A[i,i]: = 1/y; j: = Z[i]; Z[i]: = Z[k]; Z[k]: = j;
for k: = 1 step 1 until L, p step 1 until N do
  for j: = 1 step 1 until L, p step 1 until N do
    A[k,j]: = A[k,j] - B[j]×C[k] end;
for L: = 1 step 1 until N do begin k: = Z[L];
for j: = L while k ≠ j do begin
  for i: = 1 step 1 until N do begin w: = A[j,i]; A [j,i]: = A[k,i]; A[k,i]: = w end;
i: = Z[k]; Z[k]: = Z[j]; k: = Z[j]: = i end end end;

procedure Mulle(mat1,mat2,Nrad,Nmel,Nkol,mat3,fram); integer Nrad, Nmel, Nkol,
    fram; array mat1,mat2,mat3;
begin real w; integer i,j,m;
for i: = 1 step 1 until Nrad do for j: = 1 step 1 until Nkol do begin w: = 0;
  for m: = 1 step 1 until Nmel do
  if fram = 1 then w: = w + mat1[i,m]×mat2[m,j] else w: = w + mat1[m,i]×mat2[m,j];
  mat3[i,j]: = w end end Mulle;

procedure Pinus(pinne1,mat,N,pinne2,fram); integer N,fram; array mat, pinne1,pinne2;
begin real w; integer i,j;
for i: = 1 step 1 until N do begin w: = 0;
  for j: = 1 step 1 until N do
  if fram = 1 then w: = w + pinne1[j]×mat[j,i]
  else w: = w + mat[i,j]×pinne1[j];
pinne2[i]: = w end end Pinus;

  ln10: = ln(10); Rs: = 0; Tage: = false;
KNUT: read (Rurik);
  if Rurik = 1 then goto UBBE [Typ];
  if Rurik = 2 then begin write ('UTTÅG'); goto UBBE [Typ] end;
  if Rurik = 3 then goto STEG;
  if Rurik = 4 then begin read (w); goto STEG end;
  if Rurik = 5 then begin write ('skott'); goto LETA end;
```

Table 1 (*continued*)

```
          if Rurik = 6 then begin Rs: = 0 goto DATA end;
          if Rurik = 7 then goto LÄSK;
          if Rurik = 8 then begin read (start, stegbyt,tol); goto KNUT end;
          if Rurik = 9 then begin read (Typ); goto KNUT end;
          if Rurik = 10 then begin Orvar: = 0; Tage = true; goto SÄRK end;
          if Rurik = 11 then begin read (Rs1, Rs2); Orvar: = 0; goto KNUT end;
          if Rurik = 13 then goto SKRIK; comment slut KNUT;

DATA:     if Rs > 0 then goto Data 1;
          read(Ns, Nag, Nas, Nap, for i: = 1 step 1 until Nag do ag[i]);
          Orvar: = 0
Data1:    Rs: = Rs + 1; if Rs > Ns then goto KNUT;
          read (Np[Rs], for i: = 1 step 1 until Nas do as[Rs,i]);
          for Rp: = 1 step 1 until Np[Rs] do for i: = 1 step 1 until Nap do read (ap[Rs,Rp,i]);
          goto PUTS[Typ]; comment slut DATA;

GROP:     Pinus(pinne,rutinv,N,vbom,1); Uno: = Uc;
          for i: = 1 step 1 until N do Uno: = Uno − pinne[i]×vbom[i];
          if Uno > 0 then goto Grop1;
          write ('MINUSGROP');
          goto Grop2;
Grop1:    if Tage then m: = Np[Rs]
            else begin m: = 0; for i: = Rs1 step 1 until Rs2 do m: = m + Np[i] end;
          sig2y: = Uno/(m − N); sigy: = sqrt(sig2y);
          write ('Sigy = ',sigy);
Grop2:    write ('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 then ks[Rs,ik]: = kbom[i] else k[ik]: = kbom[i];
          if dia1[i] > 0 and Uno > 0 and Rurik ≠ 15 then begin w: = abs(sqrt(sig2y×dia1[i])×SH[i, i]);
            darr1[i]: = w end else w: = − 1;
          if not Tage and Rurik ≠ 15 then dark[ik]: = darr1[i];
          if dia2[i] > 0 and Uno > 0 then w1: = darr2[i]: = sqrt(sig2y×dia2[i]) else w1: = − 1;
          write(ik, kbom[i],w,w1);
          if Tage then darks[Rs,ik]: = darr2[i] else dark2[ik]: = darr2[i] end;
          if Tage then goto PROVA;
          if N = 1 and w < 0 then begin w: = sqrt(abs(0.01×Uc/pinne[1])); ik: = ivar[1]; dark[ik]: =
            abs(w×SH[1,1]) end;
          if Rurik = 15 or N = 1 then goto MIKO;
          goto VRID; comment slut GROP;

LETA:     if Orvar = 0 or Orvar = − 2 then goto UBBE [Typ];
          for i: = 1 step 1 until N do v[i]: = 0;
          if Orvar = 1 then goto Leta0;
          if Orvar = − 1 then begin
            if U > 1.5×Umin then begin
              Orvar: = − 3; goto STEG end
            else Orvar: = 1 end;
          if U > Umin then goto LetaX[Orvar];
Leta0:    Umin: = U; for i: = 1 step 1 until N do kmin[i]: = kv[i]; goto LetaX[Orvar];
Leta1:    Uc: = U;
          for i: = 1 step 1 until N do for j: = 1 step 1 until N do SH[i,j]: = s[i,j]×stek[j];
          Ri: = 1; Rj: = 0; v[1]: = 1; Orvar: = 2; goto Leta9;
```

Table 1 (*continued*)

```
Leta2:   U1: = U; v[Ri]: = − 1; Orvar: = 3; goto Leta9;
Leta3:   U2: = U; if U2 > U1 then goto Leta5;
           U2: = U1; U1: = U; stek[Ri]: = −stek[Ri];
Leta5:   pinne[Ri]: = 0.25×(U2 − U1); ruta[Ri,Ri]: = 0.5×(U2 + U1) − Uc;
         Ri: = Ri + 1; if Ri > N then goto Leta6;
         v[Ri]: = 1; Orvar: = 2; goto Leta9;
Leta4:   ruta[Ri,Rj]: = ruta[Rj,Ri]: = 0.5×(U − Uc) + pinne[Ri] + pinne[Rj] − 0.5×(ruta[Ri,Ri] +
             ruta[Rj,Rj]);
         Rj: = Rj + 1; if Rj > N then goto Leta7 else goto Leta8;
Leta6:   Ri: = 0; Orvar: = 4;
         for i: = 1 step 1 until N do for j: = 1 step 1 until N do SH[i,j]: = s[i,j]×stek[j];
         for i: = 1 step 1 until N do stek[i]: = abs(stek[i]);
Leta7:   Ri: = Ri + 1; if Ri = N then goto Leta10;
         Rj: = Ri + 1;
Leta8:   v[Ri]: = 1; v[Rj]: = 1;
Leta9:   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;
Leta11:  if Orvar = 1 then goto VÄND; goto UBBE [Typ];
Leta10:  Ri: = 0; Rj: = 0; Orvar: = 1; Rv: = N; goto Leta11; comment slut LETA;


LETAE:   begin real C, y; ik: = ivar[1]; goto LetaX[Orvar];
Leta1:   Umin: = U, Uc = U; kc[ik]: = ks[Rs,ik]; ks[ik]: = ks[ik] + stek[1]; Orvar: = 2;
         goto UBBE [Typ];
Leta2:   if Umin > U then begin Umin: = U; kmin[1]: = ks[Rs,ik] end;
         U1: = U; Orvar: = 3; ks[Rs,ik]: = ks[Rs,ik] − 2×stek[1]; goto UBBE [Typ];
Leta3:   if Umin > U then begin Umin: = U; kmin[1]: = ks[Rs,ik] end;
         U2: = U; Orvar: = 1;
         C: = 0.5×U1 + 0.5×U2 − Uc; vbom[1]: = 0.25×(U1 − U2)/C;
             kbom[1]: = ks[Rs,ik] + stek[1] + stek[1]×vbom[1];
         ks[Rs,ik]: = kbom[1]; Uno: = Uc − C×vbom[1]×vbom[1];
         if Uno < 0 then begin write ("minusgrop"); w: = − 1 end
         else begin w: = Uno/(Np[Rs] − 1); y: = sqrt(w); write ("sigZ = ",y);
         if C > 0 then begin y: = sqrt(w/C); w: = y×stek[1]; darks[Rs,ik]: =
             w else w: = − 1 end; write (kbom [1], "DARR",w);
         goto PROVA end LetaE;;


LÄSK:    begin; integer Nbyk, Nbyks, Negk, skin;
         Orvar: = 0;
         read (Nk,Nbyk);
           if Nbyk = − 1 then goto Läsk6; if Nbyk < Nk then goto Läsk1; read (Nak);
         for ik: = 1 step 1 until Nk do read (k[ik], for i: = 1 step 1 until Nak do ak[ik,i]);
         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; dark[ik]: = − 1; dark2[ik]: = − 1;
           posk[ik]: = true end;
Läsk2:   read (Nks, Nbyks);
           if Nbyks = 0 then goto Läsk5;
           if Nbyks = − 1 then goto Läsk8;
           if Nbyks = Nks then goto Läsk3;
           if Nbyks = 1 then goto Läsk4 else goto Läsk5;
Läsk4:   read (j, for Rs: = 1 step 1 until Ns do ks[Rs,j]); goto Läsk5;
Läsk3:   for Rs: = 1 step 1 until Ns do for i: = 1 step 1 until Nks do begin
           read (ks[Rs,i]); darks[Rs,i]: = − 1 end;
Läsk5:   read (skin);
         for m: = 1 step 1 until skin do
           read (ik,jk,sk[ik,jk]); goto Läsk7;
Läsk1:   for m: = 1 step 1 until Nbyk do begin
```

Table 1 (*continued*)

```
          read (ik,k[ik],for i:=1 step 1 until Nak do ak[ik,i]);
          dark[ik]:=-1; dark2[ik]:=-1 end; goto Läsk2;
Läsk6:    read (Negk);
          for m:=1 step 1 until Negk do begin read (ik); posk[ik]:=false end;
Läsk7:    if Nbyk=-1 then goto KNUT; goto SKRIK;
Läsk8:    read (m); for i:=1 step 1 until m do begin read(j);
          for Rs:=1 step 1 until Ns do ks[Rs,j]:=0 end; goto Läsk5
          end LÄSK;

MIKO:     Nimi:=0;
          for i:=1 step 1 until N do begin ik:=ivar[i];
            if Tage then ks[Rs,ik]:=kbom[i]
            else if kbom[i]<0 and posk[ik] then begin Nimi:=Nimi+1; imi[Nimi]:=i; k[ik]:=0
              end
            else k[ik]:=kbom[i] end;
          if Nimi>0 then goto Miko1;
          if Rurik=15 then goto Miko2; goto PROVA;
Miko1:    write('MIKO');
          if Rurik≠15 then begin Ucspar:=Uc; for i:=1 step 1 until N do begin
              kcspar[i]:=kc[i]; kc[i]:=kbom[i] end end;
            if Nimi=N then goto Miko2;
          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,1E-35,det,sing);
          Mulle(SHinv,ruta,N,N,N,rut3,-1); Mulle(rut3,SHinv,N,N,N,rucka,1);
          for Ri:=1 step 1 until Nimi do begin i:=imi[Ri];
            for Rj:=1 step 1 until Nimi do begin j:=imi[Rj];
            Uno:=Uno+kc[i]×kc[j]×rucka[i,j] end end;
          Uc:=Uno;
          for i:=1 step 1 until N do begin pina[i]:=0;
            for Rj:=1 step 1 until Nimi do begin j:=imi[Rj];
              pina[i]:=pina[i]+kc[j]×rucka[j,i] end end;
          for Ri:=1 step 1 until Nimi do for i:=imi[Ri] step 1 until (N-Ri) do
            begin ivar[i]:=ivar[i+1]; kc[i]:=kc[i+1]; pina[i]:=pina[i+1];
            for j:=1 step 1 until (N-Ri+1) do begin rucka[i,j]:=rucka[i+1,j];
              SH[i,j]:=SH[i+1,j] end;
            for j:=1 step 1 until (N-Ri) do begin SH[j,i]:=SH[j,i+1]; rucka[j,i]:=rucka[j,i+1]
              end end;
          N:=N-Nimi;  Pinus(pina,SH,N,pinne,1);  Mulle(SH,rucka,N,N,N,rut3,-1);  Mulle
              (rut3,SH,N,N,N,ruta,1);
          Rurik:=15; Rv:=N; goto VÄND;
Miko2:    N:=Nge; Uc:=Ucspar;
          for i:=1 step 1 until N do begin ivar[i]:=ivarge[i]; kc[i]:=kcspar[i] end
          goto PROVA; comment slut MIKO;

PROVA:    if Rurik=0 then goto Prova1;
          Rurik:=0; write ('PROVA'); goto UBBE [Typ];
Prova1:   if U<Umin then goto Prova3; if Uc>Umin then goto Prova2;
          write ('GAMLA KONSTANTER');
          U:=Uc;
          for i:=1 step 1 until N do begin ik:=ivar[i];
            if Tage then ks[Rs,ik]:=kc[i] else k[ik]:=kc[i] end; goto Prova4;
Prova2:   write ('SLUMPSKOTT');
          U:=Umin;
          for i:=1 step 1 until N do begin ik:=ivar[i];
            if Tage then ks[Rs,ik]:=kmin[i] else kc[i]:=k[ik]:=kmin[i] end;
          goto Prova4;
Prova3:   if not Tage then begin for i:=1 step 1 until N do begin ik:=ivar[i]; kmin[i]:=kc[i]:=
              k[ik] end; Umin:=U end;
```

Table 1 (*continued*)

```
Prova4:  if Tage then begin Rurik:=10; goto SÄRK end;
         goto KNUT: comment slut PROVA;

SKRIK:   write ('k(ik):');
         for ik:=1 step 1 until Nk do
           write (ik,k[ik], for i:=1 step 1 until Nak do ak[ik,i], if
             dark2[ik]>0 then begin 'DARR=', dark2[ik] end);
         write ('ks:');
         for Rs:=1 step 1 until Ns do
           write (Rs, for i:=1 step 1 until Nks do begin ks[Rs,i], if darks[Rs,i]>0 then begin
             'DARR=', darks[Rs,i] end end);
         goto KNUT; comment slut SKRIK;

STEG:    begin
         procedure Komner; begin real max, slask; integer jmax;
         w:=U/Umin -1; w:=0.1/sqrt(w); write ('KOMNER');
         for i:=1 step 1 until N do begin ik:=ivar [i];
         if kmin[i]<k[ik] then begin kv[i]:=kc[i]:=k[ik]:=kmin[i]; max:=0;
             for j:=1 step 1 until N do begin slask:=abs(s[i,j]×stek[j]);
           if slask>max then begin max:=slask; jmax:=j end end;
           stek [jmax]:=stek[jmax]×w end end end Komner;
         procedure Pluska; begin real max; 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; kv[i]:=kc[i]:=k[ik];
             if Orvar=1 then Orvar:=-2 end end Pluska;
         if Orvar=-3 then goto Steg1;
         if Rurik=4 then goto Steg4;
         read (N);
         for i:=1 step 1 until N do begin
           read (ik,w);
           ivar[i]:=ik; kv[i]:=kc[i]:=k[ik];
           if w>0 then stek[i]:=w
         else begin if dark[ik]>0 then stek[i]:=-w×dark[ik]
           else stek[i]:=0.1 end end;
Steg1:   for i:=1 step 1 until N do begin ik:=ivar[i];
           for j:=1 step 1 until N do begin jk:=ivar[j]; s[i,j]:=sk[ik,jk] end end;
           if Orvar=-3 then KOMNER;
         for i:=1 step 1 until N do begin ik:=ivar[i]; if posk[ik] then Pluska end;
         Nge:=N;
         for i:=1 step 1 until N do begin ivarge[i]:=ivar[i]; stekge[i]:=stek[i] end;
         if Orvar=-3 then begin Orvar:=-2; goto UBBE(Typ) end; goto KNUT;
Steg4:   for i:=1 step 1 until N do begin
           ik:=ivar[i]; kv[i]:=kc[i]:=k[ik]; stek[i]:=stekge[i];
           if dark[ik]>0 then stek[i]:=w×dark[ik] end;
         goto Steg1 end STEG;

SÄRK:    if Orvar>0 then goto Särk1;
         read (N);
         for i:=1 step 1 until N do begin
           read (ik,stek[i]);
           ivar[i]:=ik; end;
         for i:=1 step 1 until 12 do for j:=1 step 1 until 12 do s[i,j]:=0;
           for i:=1 step 1 until 12 do s[i,i]:=1;
         Rs:=Rs1; goto Särk2;
Särk1:   Rs:=Rs+1; if Rs>Rs2 then goto Särk3;
         if Np[Rs]≤N then goto Särk1;
Särk2:   Orvar:=0;
         write ('SATS', Rs);
```

Table 1 (*continued*)

```
          for i:=1 step 1 until N do begin ik:=ivar[i]; kc[i]:=ks[Rs,ik] end;
          goto UBBE [Typ];
Särk3:    Orvar:=0; Tage:=false; N:=Nge;
          for i:=1 step 1 until N do ivar[i]:=ivarge[i];
          goto KNUT; comment slut SÄRK;

VÄND:     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,1E-35,det,Sing);
          if Rv=N then goto GROP else goto VRID;
Sing:     write('Sing'); goto KNUT; comment slut VÄND;;

VRID:     if N=Rv then goto Vrid3;
          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;
Vrid1:    Rv:=Rv-1; if Rv=1 then goto Vrid2; goto VÄND;
Vrid3:    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; goto Vrid1;
Vrid2:    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];

SIK:      write ('SIK:');
          Mulle (SH,rut1,N,N,N,s,1);
          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];
            write (ik,jk,sk[ik,jk]) end;
          goto MIKO;

Puts1:    comment ap=log a, Z(y), lna;
          for Rp:=1 step 1 until Np[Rs] do begin ap[Rs,Rp,3]:=ln10×ap[Rs,Rp,1]; vikt [Rs,
              Rp]:=1 end;
          goto DATA; comment slut PUTS;

UBEX1:    begin real Btot, BZ, lna, lnb, lnc, steg, toly, x, x1, x2, y, y0, y1, y2, Zber; integer Karl;
          real array c, lnbeta[1:20]; integer array pot, p, q[1:20];
          switch Ada:=Ada1, Ada2, Ada3, Ada4, Ada5, Ada6, Ada7;
          comment ag= / as=Btot / ap=loga, Z(y), lna /k=betapq / ak=pot, p,q / ks=dZ;
          for ik:=1 step 1 until Nk do begin pot[ik]:=ak[ik,1];
            if k[ik]>0 then lnbeta[ik]:=ln(k[ik])+ln10×pot[ik] else lnbeta[ik]:=-1000;
              p[ik]:=ak[ik,2]; q[ik]:=ak[ik,3] end;
          U:=0;
          if not Tage then Rs:=Rs1;
Ubbe1:    Btot:=as[Rs,1]; y0:=Btot; toly:=tol×y0;
          if Rurik=2 then write('SATS', Rs, 'BTOT=', Btot);
          Rp:=0;
Ubbe2:    Rp:=Rp+1; if Rp>Np[Rs] then goto Ubbe3;
          lna:=ap[Rs,Rp,3];
SLINGA:   steg:=1; Karl:=1; x:=start;
Tjat:     lnb:=x; if lnb<-23 then y:=0 else y:=exp(lnb);
          for ik:=1 step 1 until Nk do
            begin lnc:=p[ik]×lna+q[ik]×lnb+lnbeta[ik];
            if lnc>2 then begin y:=8; goto Ada[Karl] end else if lnc<-23 then c[ik]:=0 else
            c[ik]:=exp(lnc);
            y:=y+q[ik]×c[ik] end; comment y=Bber;
          w:=abs(y-y0); if toly>w then goto Nog; goto Ada[Karl];
Ada1:     if y>y0 then begin Karl:=3; x:=x-2 end
            else begin Karl:=2; x:=x+2 end; goto Tjat;
```

Table 1 (*continued*)

```
Ada2:   if y>y0 then begin Karl:=4; x:=x-steg end else x:=x+2; goto Tjat;
Ada3:   if y>y0 then x:=x-2 else begin Karl:=4; x:=x+steg end; goto Tjat;
Ada4:   steg:=0.5*steg; if steg<stegbyt then goto Byt;
        if y>y0 then x:=x-steg else x:=x+steg; goto Tjat;
Ada5:   if y>y0 then goto Byt; y1:=y; Karl:=7; goto Korda;
Ada6:   if y>y0 then begin y2:=y; Karl:=7; goto Korda end; goto Byt;
Ada7:   if y>y0 then begin x2:=x; y2:=y end else begin x1:=x; y1:=y end;
Korda:  w:=(x2-x1)/(y2-y1); x:=x1+w*(y0-y1); goto Tjat;
Byt:    if y>y0 then begin x2:=x; y2:=y; x:=x-steg; x1:=x; Karl:=5 end
          else begin x1:=x; y1:=y; x:=x+steg; x2:=x; Karl:=6 end;
        goto Tjat;
Ubbe3:  if Tage then goto Ubbe4;
        if Rs2>Rs then begin Rs:=Rs+1; goto Ubbe1 end;
Ubbe4:  for i:=1 step 1 until N do begin ik:=ivar[i];
          write(if Tage then ks[Rs,ik] else k[ik]) end;
        write ('U=',U);
        if Orvar=0 then Orvar:=1;
        if Orvar=-2 then Orvar:=-1;
        if Rurik=0 then goto PROVA;
        if Rurik=1 or Rurik=2 then goto KNUT
        if Tage and N=1 then goto LETAE; goto LETA;
Nog:    BZ:=0; for ik:=1 step 1 until Nk do BZ:=BZ+p[ik]*c[ik]; Zber:=BZ/Btot;
          w:=Zber-ap[Rs,Rp,2]+ks[Rs,1]; U:=U+w*w*vikt[Rs,Rp];
        if Rurik=2 then goto UTTÅG else goto Ubbe2;
UTTÅG:  write(ap[Rs,Rp,1], ap[Rs,Rp,2],1000*w);
        goto Ubbe2; end UBBE;

Puts2:  write ('tom');
UBEX2:  write ('tom');
        end LETAGROPVRID
```

It should be noted that in the basic input blocks DATA and LÄSK the instructions for reading data and preliminary constants are given in a very general form so that they are applicable to practically any type of problem.

Function of the blocks

In the block diagram, Fig. 1, the full-drawn lines indicate the course of the computer's work, after reading some specific number *Rurik* in KNUT. For instance, if *Rurik* = 6 has been given, the computer uses the blocks DATA and PUTS to read the data, and do some preliminary calculations. If *Rurik* = 5, it starts a "shot", which eventually leads it through most of the blocks to the right of KNUT in Fig. 1.

The broken lines indicate the course when the group constants *ks* are varied (*Rurik* = 10, *Tage* = **true**, SÄRK in control). The dotted lines finally are followed when MIKO has been set in action.

The numbers are *Rurik* if nothing else is stated.

Rurik values in parentheses are given internally usually at the place indicated. So is "(*Orvar*: = 1)". The other statements, "*Nbyk* = −1" etc. (like the *Rurik*) are conditions for choosing the path so marked.

The "blocks" of Fig. 1 are logical units. As written in Table 1, some of them are blocks in the strict ALGOL sense, some are not but could easily be made so by adding one **begin**, one **end**, and some declaration at the proper places.

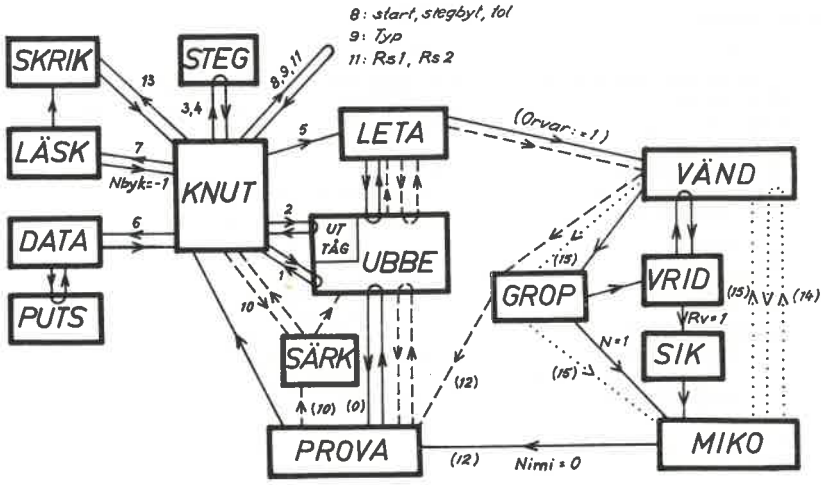

Fig. 1. Block diagram of LETAGROP program, with VRID and MIKO. It corresponds to the program in Table 1 with one exception: In Fig. 1 the program passes from MIKO to VÄND (Rurik: =14) for a matrix inversion, in Table 1 the inversion is made inside MIKO.

Input blocks

DATA reads the experimental input data, and PUTS (a special block) may use these data to calculate some quantities which will be needed for the further calculations and which are independent of the unknowns k and ks.

LÄSK (*läs konstanter*) reads the preliminary values for the constants (k and ks) to be determined.

STEG (steps) reads the general orders for the "shot" (the systematic variation of constants): which constants are to be varied, and by how much. Before starting a "shot", STEG also checks, by means of the procedure PLUSKA, that no constant of those that must be non-negative will become negative in the course of the variation.

SÄRK (*särskilda konstanter*) takes control, instead of STEG, in case the specific constants ks for each group are to be varied.

The remaining blocks are concerned with calculations and output.

Other blocks

UBBE (=*U beräknas*) is a special block which calculates U, for each set of constants given.

LETA (search) governs the systematic variation of the N unknown constants. Using the values for U successively obtained from UBBE, it calculates the terms in the vector *pinne* (**p**) and the square matrix *ruta* (**R**), which are the coefficients in the second-degree equation for U.

GROP (pit) uses *ruta* and *pinne* to calculate the position of the minimum (*kbom*) and the distances (*dark*) to the D boundary, and thus obtain $\sigma(y)$, $\sigma(\mathbf{v})$ and $\sigma(\mathbf{k})$.

VÄND (invert) uses the matrix inversion procedure INVERT, which is taken from the library of the Swedish computer BESK.

VRID (twist) + SIK calculate a new twist matrix.

MIKO (*minuskonstanter*) tests whether any of the calculated values *kbom* are negative although the constants must be positive or zero. If so these constants are set = zero; for the remaining constants, MIKO + GROP calculate those values for *kbom* and *dark* that would have been obtained, had the "minus" constants been exactly zero. MIKO has to be repeated, if some of the new *kbom* turn negative.

PROVA (test) uses UBBE to calculate U for the set of constants *kbom* calculated by GROP (and MIKO). If the new set gives a lower U than earlier, it is accepted as the central set. If the previous central set gave a lower U, it is retained, and "gamla konstanter" (= old constants) is written. If one of the sets used in LETA or some earlier set gave a still lower U, this is used ("slumpskott" = hit by accident).

SKRIK (*skriv konstanter*) writes the "best" values for the constants that the computer has at a given moment, and their standard deviations if available.

MULLE and PINUS are procedures for matrix × matrix and vector × matrix multiplication.

Symbols in basic blocks

As a rule, our symbols in the ALGOL program contain several letters. Non-Scandinavians may consider them as convenient arbitrary symbols; to Scandinavians many of them (and also the names of blocks and labels) may give helpful associations. "(III:27)" etc. refer to numbers of equations in part III [3].

Booleans

Tage is a Boolean variable which is **true** when the separate constants *ks* for each group are varied ("en sats i tage"), otherwise **false**, especially when the common constants *k* are varied.

Posk[*ik*] is **true** if *k*[*ik*] must be positive or zero, **false** if *k*[*ik*] is allowed to be negative.

Control numbers

Rurik is read at KNUT. Each integer value for *Rurik* between 1 and 11, or 13, corresponds to a specific task to be carried out (see below for details). *Rurik* = 0, 12, 14, 15 are given internally to govern the path between UBBE and PROVA or MIKO and VÄND.

Orvar = 1 means that a central set of constants k_c and a corresponding value U_c for U are available so that a systematic variation ("shot") can be started. To avoid starting the variation around an erroneous value for U_c, *Orvar* is set = 0 every time the data or the central constants are changed. *Orvar* = 2, 3 and 4 are used to govern the function of LETA.

Orvar is set equal to −2 when the values for k_c have been changed by the procedure PLUSKA and the new U_c has not yet been calculated, but *Umin* and *kmin* are available. When the new U_c has been calculated *Orvar* is changed from −2 to −1 and then to 0. *Orvar* = −3 is used in connection with the procedure KOMNER (= come down), when the values for *kc* at first calculated by PLUSKA have given too high a value for the U.

Typ may be used in case one wants to treat several types of problems without changing the main program: PUTS and UBBE are then switches, governed by *Typ*.

For changing to a new type of problem one gives *Rurik* = 9 and the new value for *Typ*, after which the program switches over to the corresponding versions of UBBE and PUTS. (In Table 1, only dummies are given for *Typ* ≠ 1.)

Fram in procedures Mulle and Pinus is set = 1 for normal multiplication, otherwise one matrix is transposed.

Other integers, and integer arrays

i, *j* = serial number of constant among *k* or *ks* to be varied;
ik, *jk* = serial number of constant among all the *k* or *ks*;
ip = serial number of point within its group;
imi[1:*Nimi*] = index *i* of "minus" constant *kbom*[*i*];
ivar[*i*] = number *ik* of *i*th constant to be varied;
ivarge[*i*] = *ivar*[*i*] for common constants, to be saved during *ks* variation or MIKO;
m = integer in general;
N = number of constants to be varied, either common or group constants;
Nag, *Nak*, *Nap*, *Nas* = number of quantities *ag*, *ak*, *ap* and *as*;
Nbyk, *Nbyks* = number of *k* and *ks* values to be changed after reading *Rurik* = 7;
Negk = number of *k* values for which *posk*[*ik*] will be set **false** (removing the mechanism PLUSKA that will keep *k* positive or zero);
Nge = *N* for common constants, to be saved during variation of *ks* (SÄRK) or MIKO;
Nimi = number of "minus" constants among the *kbom*[i];
Nk, *Nks* = total number of *k* and *ks*;
Np[*Rs*] = number of points in group no. *Rs*;
Ns = number of groups;
Ri, *Rj* = indices for *kv*[*Ri*] and *kv*[*Rj*] to be varied at a certain step of LETA;
Rp = serial number of point within its group;
Rs = serial number of group studied;
*Rs*1 and *Rs*2 = serial numbers of first and last group to be treated in calculation;
Rv = dimension of square matrix during stepwise calculation of correction matrix *vri* (**W**);
skin = number of *sk* elements to be read after *Rurik* = 7.

Real variables

det = determinant in INVERT;
*ln*10 = 2.3026;
sigy = $\sigma(y)$, standard deviation in *y* (here Z) (I:17);
sig2y = $\sigma^2(y)$ (I:17);
start = first value to be given to unknown to be eliminated (here *lnb*) in UBBE;
stegbyt = value for *steg* in SLINGA at which chord shooting should replace binary approach;
tol = relative tolerance allowed in solving the equation in SLINGA. (In case no equations need to be solved before calculating *U*, then *start*, *stegbyt* and *tol* may be used for other purposes.)
U = error-square sum, or other function to be minimized, calculated in UBBE (III:1, IV:4);
Uc = *U* for central set *kc* (U_c, III:7 or U_c' in reduced pit, III:39);
Ucspar = U_c to be saved during MIKO;
Umin = lowest value found for *U* during systematic variation;

$U1$, $U2 = U_1$ and U_{-1}, (III:8);
Uno = calculated U at minimum of second-degree surface (U_o, III:11, 12b);
w = numerical value in general.

Arrays

In arrays, *Rs* refers to the serial number of a group, *Rp* to the number of a point within the group, and *ik* and *jk* to the number of a constant, *k* or *ks*, among all the constants. The numbers *i* and *j* refer to their place within the set of constants to be varied. They are connected by $ik = ivar[i]$ and $jk = ivar[j]$. In the program, *i* and *j*, like *m*, are sometimes used also as general counting indices.

Sometimes the dimensions of an array are given, sometimes the general index. *Rp* is supposed to run from 1 to $Np[Rs]$, *i* and *j* from 1 to *N*, *ik* and *jk* from 1 to *Nk*, and *Rs* from 1 to *Ns*.

$ag[1{:}Nag]$ = quantities common to all points;
$ak[1{:}Nk, 1{:}Nak]$ = quantities belonging to each constant *k*;
$ap[1{:}Ns, 1{:}Np[Rs], 1{:}Nap]$ = experimental quantities for each point;
$as[1{:}Ns, 1{:}Nas]$ = quantities common to all points in each group of data;
$dark[ik] = darr1[i]$ for common constant, saved for estimating *stek*, set as -1 until calculated;
$dark2[ik] = darr2[i]$, standard deviation $\sigma(k[ik])$, set as -1 until calculated;
$darks[1{:}Ns, 1{:}Nks]$, standard deviation σ for *ks*;
$darr1[i]$ = distance to D boundary, ($h_i\sigma(v_i)$, III:27, Fig. 2);
$darr2[i]$ = standard deviation σ for $kbom[i]$ ($\sigma(k_i)$, III:28, 30, Fig. 2);
$dia1[i]$ = diagonal term in $\mathbf{R}^{-1}$;
$dia2[i]$ = diagonal term in $\mathbf{SHR}^{-1}(\mathbf{SH})^{\mathrm{T}} = \mathbf{A}^{-1}$ (III:30);
$k[ik]$ = "unknown" constants, common to all points;
$kbom[i]$ = value for constant at minimum calculated in GROP ($\mathbf{k}_o$, III:13);
$kc[i]$ = central value for constant *k* (or *ks*) during variation ($\mathbf{k}_c$ or $\mathbf{c}$, III:4);
$kcspar[i] = kc$ for common constants *k*, saved during variation of *ks* or MIKO;
$kmin[i]$ = constant from set giving lowest *U* found, *Umin*;
$ks[1{:}Ns, 1{:}Nks]$ = "unknown" constants special for the various groups;
$kv[i]$ = actual value for *k* or *ks* at some step of the variation;
$pina[i]$ = auxiliary vector $\vec{\mathbf{b}}_-\ \mathbf{A}_{-+}$ (III:39);
$pinne[i]$ = vector $\mathbf{p}$ for linear terms in equation $U(\mathbf{v})$ (III:7) or vector $\mathbf{p}'$ for reduced pit (III:38, 39);
$rucka[1{:}N, 1{:}N]$ = matrix $\mathbf{A}$ of second-degree terms in $U(\mathbf{k})$ (III:35, 35a);
$ruta[1{:}N, 1{:}N]$ = matrix of second-degree terms, $\mathbf{R}$, in $U(\mathbf{v})$ (III:7) or $\mathbf{R}'$ for reduced pit (III:38, 39);
$rutinv[1{:}N, 1{:}N$, or $1{:}Rv, 1{:}Rv]$ = inverted matrix $\mathbf{R}^{-1}$, in VRID also inverted submatrices, $\mathbf{R}_{Rv}^{-1}$ etc. (III:19a);
$rut1[1{:}N, 1{:}N]$ = auxiliary matrix, $\mathbf{SHR}^{-1}$ or $\mathbf{WH}^{-1}$;
$rut3[1{:}N, 1{:}N]$ = auxiliary matrix, $((\mathbf{SH})^{\mathrm{T}})^{-1}\ \mathbf{R}$ or $(\mathbf{SH})_{++}{}^{\mathrm{T}}\ \mathbf{A}_{++}$;
$s[1{:}N, 1{:}N]$ = matrix $\mathbf{S}$ (III:6);
$SH[1{:}N, 1{:}N]$ = matrix product $\mathbf{SH}$ (III:4, 5, 6);
$SHinv[1{:}N, 1{:}N] = (\mathbf{SH})^{-1}$;
$sk[1{:}Nk, 1{:}Nk]$ = storage for elements in $\mathbf{S}$ such that $sk[ik, jk] = s[i, j]$. If one changes the *ivar*, $sk[ik, jk]$ should sometimes be changed, to be strict. We have not thought it worth-while to introduce the necessary complications.

stek[*i*] = step to be used for the constants during systematic variation (h_1, III:5);
stekge[*i*] = *stek*[*i*] for common constants to be saved during variation of *ks*;
v[*i*] = variation vector, elements usually set equal to +1,0, or −1 (v, III:4);
vbom[*i*] = values for *v*[*i*] at calculated minimum point (v_o, III:12a);
vikt[1:*Ns*, 1:*Np*] = weight of experimental point, set = 1 in the present program; it can easily be given specific values, either read or calculated within the program.
vri[1:*N*, 1:*N*] = correction matrix **W** (III:14, 19a).

Symbols in special blocks (PUTS and UBBE) for Z problem

Btot = total concentration of reagent B, *B*;
BZ = total concentration of A bound, *BZ*;
lna = ln*a*, where *a* = [A] is the concentration of free reagent A;
lnb = ln*b*, where *b* = [B] is the concentration of free B;
lnc = ln of concentration of complex no. *ik*, $[A_pB_q]$;
c[*ik*] = concentration of complex;
lnbeta[*ik*] = ln β_{pq} for complex;
Karl = control number for approach to the *lnb* value that solves eqn. (1);
p[*ik*], *q*[*ik*] = integers in formula of complex A_pB_q;
pot[*ik*] = log β_{pq} − log(*k*[*ik*]), usually the expected integer part of log β_{pq}, introduced to keep *k*[*ik*] roughly within the order of magnitude, 0.1 to 1.
steg = amount by which *x* is changed in binary approach;
toly = tolerated deviation of *y* from *y0* = *Btot* in solving equation (1);
x = independent variable (here *lnb*) in SLINGA;
y = dependent variable in SLINGA, not same as *y* in (I:1) and (III:1);
y0 = required value for *y* (here *Btot*);
x1, *y1* = coordinates of lower point in chord shooting;
x2, *y2* = coordinates of upper point in chord shooting;
Zber = calculated value for *Z*.

Input, and details on input blocks

As the program is written in Table 1, UBBE and PUTS are switches, governed by *Typ*. Hence the first input must be the two integers, 9 (*Rurik*) and 1 (*Typ*).

The following input consists of two parts, as in earlier versions of LETAGROP. The first part contains the experimental data, "*in-data*" below. The *in-data* (starting with the number 6, which is *Rurik*) will in general be used for a number of consecutive cycles of calculation. The second part is "dagens spaning" (DS) (day's orders for searching), which contains various *Rurik* values, each followed by the information required; the starting values for the *k* and *ks*, orders on how to vary them, etc. The principle may be clear from the text and example below.

Data

The block DATA allows the computer to read data of varying nature.

The *in-data* are divided into *Ns* groups ("satser"). In common to all the groups ("gemensamma") are *Nag* known quantities *ag*[*i*] and in addition *Nk* unknown constants *k*[*ik*].

In each group ("sats"), for instance group no. *Rs*, there are *Np*[*Rs*] points. In common to all the points of the group ("satsegna") are *Nas* known quantities

as[*Rs*, *i*] and *Nks* unknown constants *ks*[*Rs*, *ik*]. Peculiar to each point ("punktegna") are *Nap* known quantities *ap*[*Rs*, *Rp*, *i*], one of which may be identical with the quantity *y* that is supposed to carry the error. The *in-data* are given as follows (after *Rurik* = 6):

(Common:) *Ns*, *Nag*, *Nas*, *Nap*, $(ag)_{Nag}$;

(for each group:) (*Np*, $(as)_{Nas}$, $((ap)_{Nap})_{Np})_{Ns}$.

The subscript indicates the number of values to be given.

After reading each group of data, the computer uses the special block PUTS to calculate weights and other quantities that are needed for the following calculations and are independent of the unknown constants.

Läsk

To each unknown constant *k*[*ik*] belong *Nak* special numbers *ak* ("konstantegna"), for instance *pot*, *p* and *q* (*Nak* = 3) to an equilibrium constant β_{pq}.

LÄSK is called by *Rurik* = 7. The numbers to be given then will be symbolized by *in-läsk*; they consist of three independent groups.

1. Common constants *k* ("gemensamma"). New problem: *Nk*, *Nk*, *Nak*, (*k*, $(ak)_{Nak})_{Nk}$. Partial change (or addition) of *Nbyk* constants: *Nk*, *Nbyk*, (*ik*, *k*,$(ak)_{Nak})_{Nbyk}$. No change: *Nk*, 0.

2. Group constants *ks* (satsegna). New problem: *Nks*, *Nks*, $((ks)_{Nks})_{Ns}$. If *Nks* = 1 then: 1,1, $(ks)_{Ns}$. Exchange of one of them, *ks*[*Rs*, *j*], in all sets: *Nks*, 1, *j*, (*ks*[*Rs*, *j*])$_{Ns}$. *m* of them, *ks*[*Rs*, *j*] set equal to zero in all sets: *Nks*, −1, *m*, $(j)_m$. No change: *Nks*, 0. The second integer is "*Nbyks*" in the program (number of *ks* to be changed). No provision has as yet been made for values other than *Nk*, 1 or 0; but this can easily be done should the need arise.

3. Elements for twist matrix ("vridtermer"). Some of the *sk*[*ik*, *jk*] are known and given to the computer: *skin*, (*ik*, *jk*, *sk*)$_{skin}$. Note that *jk* > *ik*. No change: 0.

Removal of safeguard. The Boolean *posk*[*ik*] if **true** prevents *k*[*ik*] from becoming negative during any calculation. This safeguard is automatically made **true** for all *k*[*ik*] but can be removed for *Negk* of the constants by calling LÄSK (*Rurik* = 7) and giving: *Nk*, −1, *Negk*, $(ik)_{Negk}$. After that the computer goes back to KNUT.

Every time one calls for *Rurik* = 7, (except for changing the *posk*), the computer passes on from LÄSK to SKRIK (*skriv konstanter*) and prints the values it has for the *k* and *ks*, with their standard deviations (*darr2*) if calculated.

Steg

Of the *Nk* unknown constants, *N* will be varied. Each constant has one number, say *ik* (from 1 to *Nk*) among all the unknown constants, and another number, say *i*, among the *N* constants to be varied. The relationship is given by the function *ivar*: *ik* = *ivar*[*i*], *jk* = *ivar*[*j*] etc. If we vary *k*[2], *k*[3], *k*[4] and *k*[6] out of *Nk* = 6 constants, then *N* = 4, 2 = *ivar*[1], 3 = *ivar*[2], 4 = *ivar*[3] and 6 = *ivar*[4].

In the first systematic variation of the unknown constants, the steps *stek*[*i*] are given in DS. When the computer has made a shot and calculated a pit of the right shape, it obtains for each constant a value *dark*[*ik*] (from Swedish "darr"), which gives the distance ($h_i\ \sigma(v_i)$, III: Fig. 2) from the pit to the D boundary along the twisted coordinate axes. By giving *Rurik* = 4 one may then order the computer to make the various *stek*[*i*] equal to a certain fraction (*w*) of the corresponding *dark*. In DS the numbers for STEG (*in-steg*) are as follows:

(*Rurik* = 3:) N, $(ivar[i], w)_N$. A positive w gives $stek[i]$ directly, a negative w makes $stek[i] = -w \times dark[ik]$. If no *dark* has been calculated, the computer takes $stek[i] = 0.1$.

(*Rurik* = 4:) w. For all $ik = ivar[i]$, $stek[i]$ is set equal to $w \times dark[ik]$ if $dark[ik] > 0$; otherwise no change.

If there is a risk that some k that must be positive will get a negative value during the variation, then the central value for that k is adjusted in STEG (procedure PLUSKA).

Särk

is called by *Rurik* = 10, which makes the Boolean *Tage* **true**. The group constants, *ks* (or some of them) will then be varied for each group separately and "better" *ks* values calculated. This variation is controlled by SÄRK instead of by STEG; *stek* and *ivar* apply to the *ks*.

In general there are only one or two *ks* per group, say an analytical error, and a value for an emf constant, E_0. It is usually satisfactory to use the same step for varying a certain *ks* in all groups; a variation of the steps, if desirable, can be achieved by using alternatingly *Rurik* = 11 and *Rurik* = 10. Since various groups give different *darr* for the corresponding *ks*, no attempt has been made to adjust the *stek* for *ks* as done for the common k with *Rurik* = 4. Nor is a twist matrix introduced for the *ks*.

The necessary information for DS (*in-särk*) will be:

(*Rurik* = 10:) N, $(ivar[i], stek[i])_N$.

For the simple case that there is only one *ks* (as in the case considered) we have made a shorter block, LETAE, which does the (not too difficult) job of LETA, GROP and VÄND.

The use of Rurik

After each *Rurik* value, below, the additional input needed (if any) is indicated, and then the task is described.

Rurik = 1: The function to be minimized, U, is calculated for groups no. *Rs1* through *Rs2*, using data, k and *ks* values available.

Rurik = 2: U is calculated; for each point quantities of interest (such as the errors) are printed (UTTÅG).

Rurik = 3, *in-steg*: N,$ivar[i]$ and $stek[i]$ are read. If necessary, procedure PLUSKA in STEG is applied to change the *kc* values.

Rurik = 4, *w*: $stek[i]$ are recalculated for the k using the *dark* available. PLUSKA is applied if necessary.

Rurik = 5: one "shot". If a set of constants and the corresponding U value are already available, after *Rurik* = 1 or 2, or from an earlier shot, the computer uses this as the central set. Otherwise, U is first calculated for the central set.

Rurik = 6, *in-data*: data are read.

Rurik = 7, *in-läsk*: k and *ks* are read or changed, and available k and *ks* are printed (except when *Nbyk* = −1).

Rurik = 8, *start*, *stegbyt*, *tol*: In the Z problem, "*start*" is the first guess for the unknown variable x (here, *lnb*), *stegbyt* determines the value for *steg* where one passes from binary approach to chord shooting, and *tol* is the tolerance when solving (1). For other problems, these quantities may be used in a different way.

Rurik = 9, *Typ*: new type of problem.

Rurik = 10, *in-särk*: variation and "improvement" of group constants *ks*.

Rurik = 11, *Rs1*, *Rs2*: *Rs1* and *Rs2* are the serial numbers of the first and last group to be treated.

Rurik = 13: available "best" values for k and ks are printed with their "darr" (SKRIK).

Rurik = 0, 12, 14 and 15 are given inside the computer for special purposes, and not from the outside.

Special problem (the "Z problem")

The special problem treated by UBBE and PUTS in Table 1 has already been defined by equations (1–4). The correspondence between the general terms and the quantities in the special problem is summarized in the "comment" in UBBE (see UBEX 1). There is no common quantity *ag* (*Nag* = 0). The data are arranged in groups of constant B. For each group there is one "known" quantity, $as[Rs,1] = B$, and one unknown constant, $ks[Rs,1] = \delta Z$, the analytical error in Z. There are three data for each point, $ap[Rs, Rp, i]$ namely $\log a$, Z, and $\ln a$; the first two are read (*Nap* = 2), the third is calculated in PUTS. The common constants $k[ik]$ correspond to the β_{pq}, and to each of them belong three integers (*Nak* = 3): $ak[ik,1] = pot[ik]$, $ak[ik,2] = p[ik]$ and $ak[ik,3] = q[ik]$.

The relationship is

$$\beta_{pq} = k[ik] \times 10^{pot[ik]} \tag{5}$$

The weights *vikt* are set equal to unity. In studies of this type, there is usually at the end an abundance of data points, and there are relatively more points for some concentrations than for others. What corresponds to a weighting is the final decision how the data for LETAGROP treatment, perhaps 150 points out of 500, should be selected. In order to have an even distribution over the B values one may e.g. decide to use every point for some B, 2 out of 3 for another, 1 out of 4 for a third etc. The selection in each group is, of course, made at random.

In cases where data are scarce, one may give weights with the data, or have the weight calculated in UBBE or PUTS.

Slinga

UBBE contains, after "SLINGA", a system of loops which solve the equation (1) for b. One could easily separate this part of the program from the rest of UBBE as a special procedure, "SLINGA" and this may sometimes have advantages.

Immediately after Tjat, an increasing function $y(x)$ is defined; in the present case it is *Btot*(*lnb*). The following calculations aim to find the value x that makes $y = y0$, a value given at Ubbe1. Most of SLINGA can be used for any increasing function $y(x)$. For a new problem one needs only to rewrite the first few lines, which define $y0$ and y, and the text after *Nog*, which tells what to do after the equation has been solved.

We have found it convenient to calculate the concentrations for individual complexes by means of logarithms, as is seen on inspection of the program after "Tjat".

Instead of the purely binary approach to the solution in our earliest program, we now use a mixture of the binary approach and "chord shooting". The shift is made when the step in *lnb* is smaller than the given quantity, *stegbyt*. The step, *steg*, starts with 2 and then becomes 1, 0.5, 0.25, 0.125, etc. The optimum value for

stegbyt depends on the degree, q, of the dominant term; low optimal values for *stegbyt* correspond to high degrees, and a rough rule might be, *stegbyt* $\approx 1/q$. The time of calculation is not very sensitive to changes in *stegbyt* by a factor of 2 or 4.

Numerical example

In Tables 2 and 3 are given extracts from a calculation on $Z(\log a, B)$ data for Ni hydrolysis in the medium 3 M (Na)ClO_4. The data were supplied by our friend Kim Aleksandrovič Burkov, from Leningrad University, and will be published elsewhere in full. As the program is written in Table 1, one must first of all give 9 (*Rurik*), 1 (*Typ*). The *in-data* begin with 6(*Rurik*), 5(*Ns*, number of groups), 0(*Nag*), 1(*Nas*), 2(*Nap*). Then follow the five groups. The first begins with 22 (*Np*[1], number of points), 0.1 (*Btot*), and then come the points: 5.907 (log $a = -\log h$), 0.000461 (Z); 6.401 (log a), 0.000560 (Z), etc., 20 more pairs, then 26 (*Np*[2]), 0.2 (*Btot*); 5.547 (log a), 0.000241 (Z) etc.

In this special case, the data had already been analyzed once by the computer. The results of the first shot, before anything was known about S, looked much like that in Table 6, part II [2]; each constant being varied in turn, and U being calculated for each set. From this shot, we obtained "better" values for $k[ik]$, and preliminary values for the elements of the twist matrix and the $\sigma(k[ik])$. A preliminary shot also gave approximate values for the group constants *ks* ($=\delta Z$). This information has been inserted in the new "dagens spaning" (Table 2): for instance, the *stek* are roughly one-third of the standard deviations found.

One reason for interrupting after the first "shot" is that one wants to give an "UTTÅG" (*Rurik* = 2), and thus to have the data for all points, and their deviations printed. Printing errors in the *in-data* will then stand out and can be corrected so that they will not influence the succeeding calculations.

The output is seen in Table 3. First come the given k and *ks*. In this case three complexes $Ni_q(OH)_p$ were assumed with $(p, q) = (1,1)$, (1,2) and (4,4).

Then come the sets of values for the three constants ($10^{10}\beta_{11}$, $10^{9}\beta_{12}$, $10^{27}\beta_{44}$) used during the "shot", and the U calculated for each set. Obviously it did not matter that $k[1]$ was first set as zero since it was increased to a positive value by PLUSKA. The first set is *kc*: then come three pairs of sets where $k[1]$, $k[2]$ and $k[3]$ are varied, and finally $k[1]+k[2]$, $k[1]+k[3]$, and $k[2]+k[3]$ are varied. A change in $k[3]$ brings with it changes in the other two, a change in $k[2]$ changes $k[1]$ but not $k[3]$, and a change in $k[1]$ does not affect the others; this is as required by equations (4–6) in part III. He who wants may check that the numerical values are in accordance with these equations, and with the s_{ij} and h_i (*stek*) given.

The calculations in GROP give $\sigma(Z)$ ("*sigy*"), and the position of the calculated minimum: i, *kbom*[i], *darr1*[i] ($= h_i\sigma(v_i)$) and *darr2*[i] ($=\sigma(k[ik])$). For the difference between the latter two quantities, see part III, Fig. 2, and the corresponding text.

The twist matrix ("sik") comes out to have practically the same elements as were found in the preliminary calculation, and given in "dagens spaning". Since $k[1]$ came out negative, MIKO is applied, and the best values for $k[2]$ and $k[3]$ in the reduced pit are calculated. Finally U at the minimum is calculated (PROVA); it is lower than the *Uc* earlier found, and is thus accepted.

A new "shot" gives the same result: the first complex (1,1) is again thrown out, and the minimum is found practically at the same place. There has been a slight improvement in U, but it is not perceptible with the number of digits printed.

Table 2. "Dagens spaning" for LETAGROP, Ni hydrolysis, 3 *M* $(Na)ClO_4$ medium.

Input numbers	Explanation
+7	Rurik for LÄSK
	Common constants
+3 +3 +3	Nk, Nk, Nak
+0 −10 +1 +2	k_1, ak_{11}, ak_{12}, ak_{13}
+0.3165 −9 +1 +1	k_2, ak_{21}, ak_{22}, ak_{23}
+0.4160 −27 +4 +4	k_3, ak_{31}, ak_{32}, ak_{33}
	Group constants
+1 +1	Nks, Nks
−0.20 E−3 +0.49 E−3 +0.42 E−4	ks_1, ks_2, ks_3
−0.36 E−3 +0.14 E−3	ks_4, ks_5
	Twist matrix
+3	skin
+1 +2 −1.817	ik, jk, sk_{12}
+1 +3 −12.39	ik, jk, sk_{13}
+2 +3 −11.96	ik, jk, sk_{23}
+8 −2 +0.4 +1E−7	Rurik, start, stegbyt, tol
+3 +3	Rurik for STEG, N (number of constants to be varied)
+1 +0.067	$ivar_1$, $stek_1$
+2 +0.0264	$ivar_2$, $stek_2$
+3 +0.00176	$ivar_3$, $stek_3$
+11+1 +5	Rurik, Rs_1 (first group), Rs_2 (last group)
+5	one shot
+4 +0.5 **+5**	adjustment of stek, new shot for *k*
+10+1 +1+5E−5	variation of group "constants" *ks*: Rurik, *in-särk*
+4 +0.5 **+5**	adjustment of stek, and new shot for *k*
+13 +2	Rurik for SKRIK and UTTÅG

Note. This computer has free input format and requires signs to be given between numbers. "$_{10}$" is written as E. *Rurik* numbers are given bold-face. In explanation, indices are given as subscripts instead of in brackets. Results are seen in Table 3.

Next, the *ks* values are varied separately, starting with the first group ("sats 1"). For three values of *ks*[1,1], the corresponding *U* (using the points of that group only!) are calculated, then $\sigma(Z)$, a "better" value for *ks*[1,1], and its standard deviation ("darr"). It is checked ("PROVA") and it can be seen that this was an improvement. Then the other groups are analyzed one after the other to give the best values of *ks*[*Rs*,1].

A new shot using the improved *ks* values, gives only slightly different values for *k*[2] and *k*[3]; *k*[1], as before, is thrown out. This ends the "pit mapping" asked for. Now SKRIK is called, and the "best" *k* and *ks* values are printed, each together with its standard deviation ("darr"). Finally comes the "UTTÅG", beginning with the *k* values used, and stating for each group the *Btot* value, and then for each point $-\log h = \log a$; Z; and $1000\ (Z_{calc} - Z_{exp})$.

Table 3. Output following "dagens spaning" in Table 2.

```
K(IK)
        1  .00000 E  0  -10   1   1
        2  .31650 E  0   -9   1   2
        3  .41600 E  0  -27   4   4

KS:
        1  -.200000 E  -3
        2   .490000 E  -3
        3   .420000 E  -4
        4  -.360000 E  -3
        5   .140000 E  -3

SKOTT
        .13678 E   0     .31650 E  0     .41600 E  0     U = .238943 E  -4
        .20378 E   0     .31650 E  0     .41600 E  0     U = .243899 E  -4
        .69775 E  -1     .31650 E  0     .41600 E  0     U = .235455 E  -4
        .88806 E  -1     .34290 E  0     .41600 E  0     U = .238505 E  -4
        .18474 E   0     .29010 E  0     .41600 E  0     U = .239938 E  -4
        .11497 E   0     .29545 E  0     .41776 E  0     U = .238030 E  -4
        .15858 E   0     .33755 E  0     .41424 E  0     U = .240315 E  -4
        .21806 E  -1     .34290 E  0     .41600 E  0     U = .235017 E  -4
        .47969 E  -1     .29545 E  0     .41776 E  0     U = .234542 E  -4
        .67000 E  -1     .32185 E  0     .41776 E  0     U = .237592 E  -4

SIGY =  .000444

KBOM:
        1  -.17181 E  0     .110 E   0     .180 E   0
        2   .29800 E  0     .703 E  -1     .936 E  -1
        3   .42038 E  0     .516 E  -2     .516 E  -1

SIK:
        1  2  -.1817 E  1
        1  3  -.1238 E  2
        2  3  -.1197 E  2

MIKO

SIGY =  .000444

KBOM:
        2  .24743 E  0     .538 E  -1
        3  .41975 E  0     .291 E  -2

PROVA
        .00000 E  0     .24743 E  0     .41975 E  0     U = 232927 E  -4

SKOTT
        .15074 E   0     .24743 E  0     .41975 E  0     U = .239516 E  -4
        .20568 E   0     .24743 E  0     .41975 E  0     U = .243765 E  -4
        .95804 E  -1     .24743 E  0     .41975 E  0     U = .236254 E  -4
        .86901 E  -1     .28257 E  0     .41975 E  0     U = .238376 E  -4
        .21458 E   0     .21229 E  0     .41975 E  0     U = .241644 E  -4
        .11877 E   0     .21654 E  0     .42233 E  0     U = .239768 E  -4
        .18270 E   0     .27832 E  0     .41717 E  0     U = .240253 E  -4
        .31965 E  -1     .28257 E  0     .41975 E  0     U = .235114 E  -4
        .63839 E  -1     .21654 E  0     .42233 E  0     U = .236506 E  -4
        .54936 E  -1     .25168 E  0     .42233 E  0     U = .238627 E  -4

SIGY =  .000444
```

Table 3 (*continued*).

```
KBOM:
        1  -.17184 E  0     .110 E  0     .180 E  0
        2   .29801 E  0     .703 E -1     .935 E -1
        3   .42038 E  0     .516 E -2     .516 E -2
SIK:
        1  2  -.1816 E  1
        1  3  -.1239 E  2
        2  3  -.1196 E  2
MIKO
SIGY =  .000444
KBOM:
        2  .24744 E  0     .538 E -1
        3  .41975 E  0     .291 E -2
PROVA
        .00000 E  0     .24744 E  0     .41975 E  0     U = .232927 E - 4
SATS 1
        -.20000 E -3     U = .744345 E -5
        -.15000 E -3     U = .752646 E -5
        -.25000 E -3     U = .747044 E -5
SIGZ =  .59521 E -3
KBOM =  -21273 E -3     DARR  .12690 E -3
PROVA
        -.21273 E -3     U = 743988 E - 5
SATS 2
        .49000 E -3     U = .730717 E -5
        .54000 E -3     U = .733968 E -5
        .44000 E -3     U = .739966 E -5
SIGZ =  .55165 E -3
KBOM =  .50200 E -3     DARR  .11033 E -3
PROVA
        .50200 E -3     U = .730357 E -5
SATS 3
        - - - - -
        - - - - -
SKOTT
        .15076 E  0     .24744 E  0     .41975 E  0     U = .239603 E -4
        .20570 E  0     .24744 E  0     .41975 E  0     U = .244043 E -4
        .95813 E -1     .24744 E  0     .41975 E  0     U = .236149 E -4
        .86923 E -1     .28259 E  0     .41975 E  0     U = .239299 E -4
        .21459 E  0     .21230 E  0     .41975 E  0     U = .240894 E -4
        .11878 E  0     .21658 E  0     .42233 E  0     U = .239703 E -4
        .18274 E  0     .27831 E  0     .41716 E  0     U = .240490 E -4
        .31978 E -1     .28259 E  0     .41975 E  0     U = .235845 E -4
        .63835 E -1     .21658 E  0     .42233 E  0     U = .236250 E -4
        .54945 E -1     .25172 E  0     .42233 E  0     U = .239399 E -4
SIGY =  .000445
KBOM:
        1  -.13327 E  0     .110 E  0     .180 E  0
        2   .26354 E  0     .703 E -1     .936 E -1
        3   .42077 E  0     .516 E -2     .516 E -2
```

Table 3 (*continued*).

```
SIK:
          1  2  -.1816 E  1
          1  3  -.1239 E  2
          2  3  -.1196 E  2
SIGY =    .000444
KBOM:
          3  .22432 E  0      .538 E  -1
          3  .42028 E  0      .291 E  -2
PROVA
          .00000 E  0      .22432 E  0      .42028 E  0      U=.232478 E  -4
K(IK)
          1  .00000 E  0  -10   1  1  DARR=.180 E   0
          2  .22432 E  0   -9   1  2  DARR=.538 E  -1
          3  .42028 E  0  -27   4  4  DARR=.291 E  -2
KS:
          1  -.212733 E  -3      DARR=.127 E  -3
          2   .501995 E  -3      DARR=.110 E  -3
          3   .544263 E  -4      DARR=.711 E  -4
          4  -.337781 E  -3      DARR=.791 E  -4
          5   .170035 E  -3      DARR=.551 E  -4
UTTÅG
          .00000 E  0      .22432 E  0      .42028 E  0
SATS 1    BTOT  .100000
          5.9070  .000461  -.7
          6.4010  .000560  -.6
          - - - - - - - - - - - -
          - - - - - - - - - - - -
          6.9910  .013996   .6
          6.9980  .016174  -.7
          7.0130  .018331  -.7
SATS 2    BTOT  .200000
          5.5470  .000241   .3
          etc. - - - - -
```

The example in Tables 2–3 brought with it no dramatic change since the preliminary treatment had already given fairly good values. At any rate it may give an idea of how the program works, and how complexes are eliminated by MIKO. A total view of the calculations on Ni hydrolysis will be given in a subsequent paper; it will contain additional examples of the "species selection".

Examples with $N=6$ or 7, of which we have many, give much longer tables without showing more of the principles.

Adaptation to various computers

The program given in Table 1 is written in the international computer language ALGOL and can thus be taken over by practically any modern computer.

The input and output orders are given at the right places but, as usual, will have to be written in different ways for different computers. UBBE, with its output

orders, will at any rate have to be rewritten for each type of problem. If one wants to vary the output orders in the basic blocks SKRIK and GROP, one may for instance make these blocks into switches, governed by *Typ*. Even the basic input blocks, DATA and LÄSK, may have to be varied with the problem if one's computer must have rigidly predetermined input formats; this is however unusual with modern computers.

For some computers with a limited capacity in the rapid memory, it may be necessary or desirable to split up the program into several independent programs which communicate by storing information in the drum memory, and reading from it. Another way to squeeze LETAGROP VRID into a computer with a modest memory would be to refrain from spreading out some of the large arrays and instead to read their elements from the drum when required. This, however, would make the program still harder to read than now; in the present program, each quantity is named by some understandable symbol. Whichever solution is chosen, the transformation should not be difficult to a chemist with some experience of programming the computer in question.

Those who have to use an IBM computer should be aware of the shortcomings of the IBM Algol compiler available at present (Feb. 1964); especially they must be careful to rename some identifiers (*steg*, *i*, *m* etc.) which at present are declared independently in several blocks; they are advised to use the inversion procedure INVERT as given in ALGOL (after renaming *i*, *j*, *k* etc.) and not to try to use any IBM library routine. Other computer systems seem to have much better ALGOL compilers.

Comparison with first LETAGROP

In the course of the two or three years we have had experience with LETAGROP, a number of improvements have been made. Some of these have been treated in some detail above, or in part III, and some have just been indicated. It may be worth-while to summarize here the more important changes (as we think, improvements), in comparison with the program given in part II [2].

Basic blocks

1. The twist matrix S has been introduced so that the k_i are varied along the axes of the pit. (Compare Fig. 1*b*, part III [3].) This has made possible a much more efficient treatment of skew pits.

2. The steps *stek*[*i*](h_i) are set at some prescribed fraction w—perhaps between 0.1 and 0.5—of the standard deviations along the twisted axes, $h_i\sigma(v_i) = darrI[i]$, as soon as the latter have been calculated. The adjustment of steps is important since steps which are too large give disturbances from terms of degree higher than second in $U(\mathbf{k})$, and steps which are too small give rounding errors.

3. The step for a new constant k' can be adjusted to a reasonable level by varying it alone ($N=1$). If the guessed step has been much too small, the three U values calculated sometimes, because of rounding errors, seem to lie on a curve with a maximum. In GROP the calculation of $\sigma(v)$ is then skipped, since it would give the square root of a negative number. However, a special order at the end of GROP calculates a value for *dark*[*i*] which will give a step of a more reasonable magnitude. Experience has shown that in this way reasonable starting values, and steps, have been ob-

tained for new constants by 2–3 shots even if the first guess was too low by 5–6 powers of ten.

4. A low value for U, once found, will not be forgotten. If the calculated set after a shot gives a higher U value than any previously found, the lower value is retained ("gamla konstanter", or "slumpskott" in PROVA). Also after the adjustment in PLUSKA, an earlier lower value for U is remembered.

5. Constants that must be non-negative are never allowed to become negative during the variation in a shot (procedure PLUSKA).

6. If a non-negative constant gets a minus value in the calculated minimum, it is set equal to 0 and that set of constants is calculated which gives a minimum for U in the "reduced pit" (MIKO + GROP). Thus in the test of the new "best" set of constants in PROVA, a non-negative constant will never appear with a minus value.

7. The group constants *ks* can be varied separately (SÄRK).

8. The input blocks are written in such a general way that they can accept practically any type of data (provided the computer, as is generally the case, has no restrictions on the input format).

Special blocks

1. The special block UBBE is written in a more general way than previously. As can be seen, it may be used for $Z(\log a, B)$ data for any combination of various complexes A_pB_q, and there is no need to write a special program for each combination. Programs for other types of problems: thermochemical data, emf data, etc. are also written so that they are valid for a whole class of problems, and so that all special information (number of complexes, their p and q etc.) can be given in the data or "dagens spaning".

2. The equilibrium constants are given in the form $\beta_{pq} = k[ik] \times 10^{pot[ik]}$, and the logarithms of the concentrations are calculated first. This avoids much trouble with very small and very large numbers.

3. SLINGA (the part of UBBE that solves the equation) contains a mixture of binary approach and chord shooting, and not only *tol* but also *start* and *stegbyt* can be varied. One can then easily adjust these three quantities according to one's requirements of (*a*) accuracy, (*b*) short computing time, and (*c*) avoiding dead loops because of rounding errors in SLINGA.

Future developments

The LETAGROP programs are under active development, especially along the following lines:

1. Application to other types of problem. Various intermediate editions of LETAGROP have already been applied successfully (references, see e.g. [3]) to data on ion exchange equilibria, distribution equilibria, emf measurements of central group, thermochemical data, and for some of these, special programs already exist for the last edition. Forsén *et al.* have applied LETAGROP VRID to calculations of nuclear spin coupling constants [4].

2. We are accumulating experience for the best strategy in constructing a "species selector" as indicated in part III [3]. One would then feed to the computer a small set of initial complexes, with their β_{pq}, and a list of conceivable complexes (perhaps with very rough guesses for their β_{pq}), and then let it run through those on the list once, or several times.

3. At present we have the choice of varying either the common constants k, or the group constants ks. By doing so alternatingly, one can hope to find the "best" values for both sets. In the cases we have studied up to now, the results seem independent of the order of these operations. One might imagine a case with a very strong covariance of the k and ks, and hence a slow approach to the real minimum; it would then be desirable to be able to vary k and ks simultaneously. This can be done already with the present program, but not elegantly.

SUMMARY

The paper gives a fairly full description of a recent ALGOL version (Table 1), of the general minimizing program LETAGROP VRID. A numerical example, from data on Ni hydrolysis, is given. The special parts in the program, Table 1, are written for minimizing the error-square sum U, with $Z(\log a, B)$ data on a system with polynuclear complexes A_pB_q. To switch over to another minimizing problem, one need only rewrite parts of the two special blocks PUTS and UBBE (Fig. 1). A list is made of improvements in comparison with first LETAGROP [2]; among the more important are the twist matrix (VRID) and the procedure for eliminating "minus" constants (MIKO). The necessary equations are given in Part III [3].

ACKNOWLEDGEMENTS

The working out to the present program has been supported by Statens Tekniska Forskningsråd (Swedish Technical Research Council). The National Swedish Office for Administrative Rationalisation and Economy (earlier Swedish National Board for Computing Machinery) has provided us with free computing time on Facit, Besk, Ferranti Mercury, Univac 1107, and IBM 7090. We wish to thank our friends at the KTH Department of inorganic Chemistry for pleasant cooperation. Dr. Roy Whiteker was kind enough to revise the English text.

Department of Inorganic Chemistry, Royal Institute of Technology (KTH), Stockholm 70, Sweden

REFERENCES

1. Part I: Sillén, L. G., Acta Chem. Scand. *16*, 159 (1962).
2. Part II: Ingri, N., and Sillén, L. G., Acta Chem. Scand. *16*, 173 (1962).
3. Part III: Sillén, L. G., Acta Chem. Scand. *18*, 1085 (1964).
4. Forsén, S., Åkermark, B., and Alm, T., Acta Chem. Scand., in press.

Tryckt den 17 december 1964

Uppsala 1964. Almqvist & Wiksells Boktryckeri AB