会話型データ処理—その13—
GEOCAPSでのデータ処理手順(2)
—とくに成分式の演算—

吉井守正・佐藤岱生（鉱床部）
Morimasa YOSHII　Taisei SATO

1. はじめに

前回にはデータ処理の条件設定などについて述べたので　今回は　データ処理行程の中でも　このシステムの特徴である成分式の演算について述べよう．

使用者による処理条件の設定（吉井・佐藤，1984）が終ると　第1図に示したデータ処理行程に入る．その主要部は各プログラム（例えばX-Y相関図や三角図など）によって異なる（吉井・佐藤，1983a，1983b，1984）．　しかし　それらの前処理行程についてはほぼ共通である．

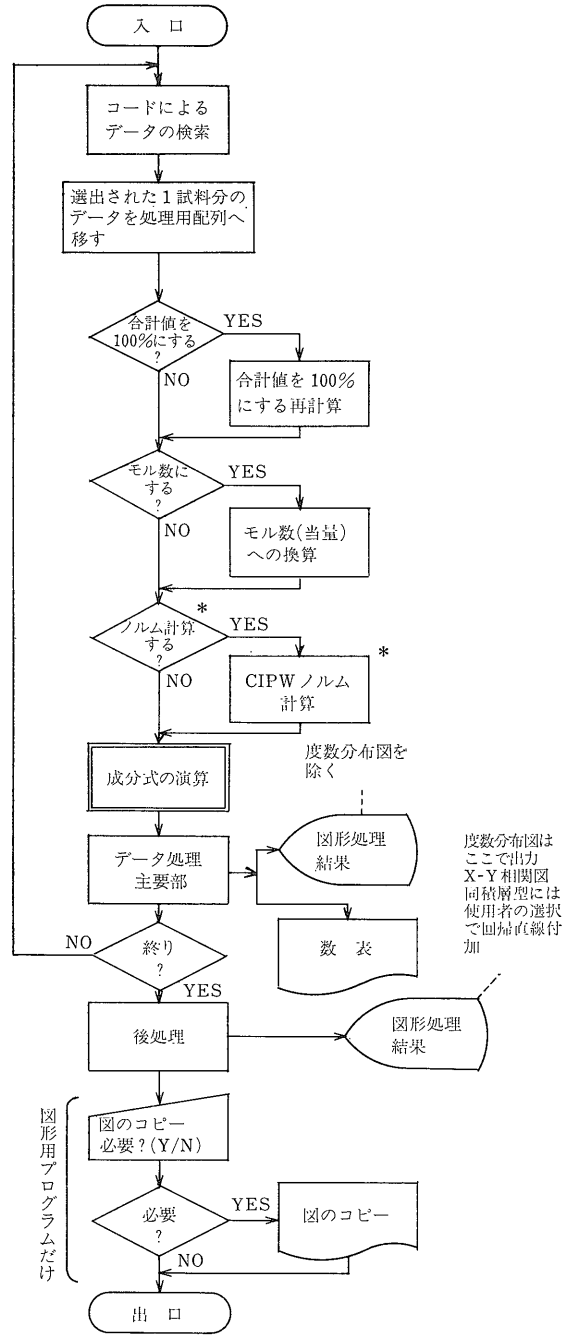
すなわち　まずコードによる検索の結果選び出された1試料分ずつのデータが　処理用配列（P）に移される．　この数値に対して　設定された処理条件に従って　合計値の100％への再計算・モル数への換算・さらには CIPW ノルム計算が必要に応じてこの配列内の数値に対して行われる．　これらの流れと配列との関係を第2図に示す．　今回は GEOCAPS で用いている変数名とその配列規模・用途などを第1表に示し　それを用いて以下の説明を行うことにしたい．

2. 成分式の演算

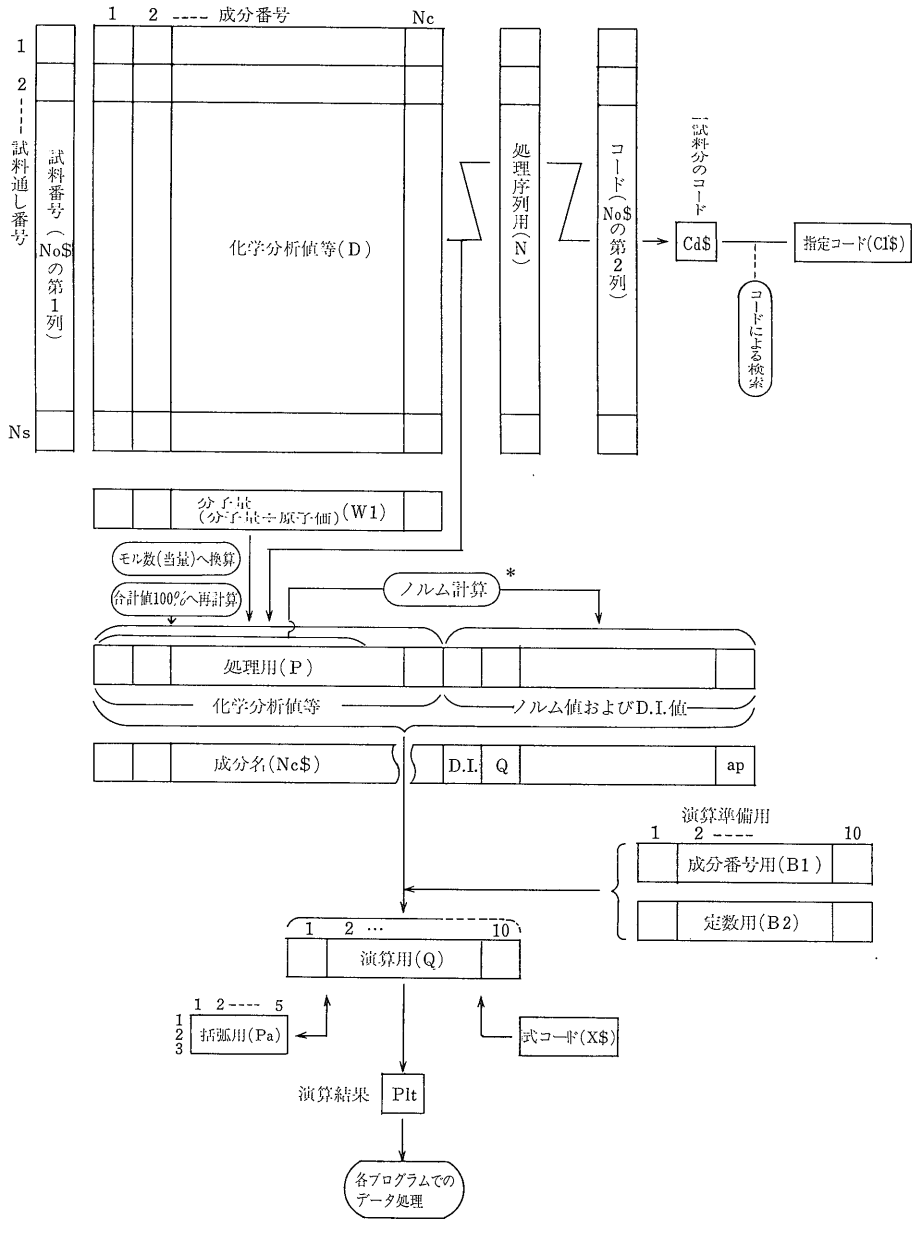
では　成分の演算式についてその原理や手順について述べよう．

a） その必要性

たとえば Fe_2O_3 と FeO から Total Fe_2O_3 または Total FeO を求めるとか　Na_2O と K_2O から Na_2O+K_2O としての処理をしたいという使用者からの要望は　初代のシステムが出来上った1977年頃からすでに出されていた．　当時はこれらの要求に応えて　その使用者専用のプログラムテープを作って配給した．　この結果特別仕様のプログラムがやたら殖えてしまった．



第1図　データ処理行程の概略的流れ図
＊ノルム計算用プログラムにだけ付属



第2図　データの検索から成分式計算までの流れと変数との関係
＊ノルム計算はノルム計算用系列のプログラムでノルム計算可能なデータタイプ（NC または NR）を取り扱う場合であって（コード別印刷用プログラムを除き）ノルム成分が処理成分として指定されたときにだけ実行される．

そこで2代目に当たる岩石化学データ処理システム（吉井・佐藤，1981）では　全鉄を求める計算と任意の2成分の和が算出できるようになった．　しかし使用者の要求はさらに増し　A＋B＋C中のA成分の比率のような算式をはじめ　多岐に亘った．　当時の対策としては必要な算式を臨時にプログラムに書き込んで実行させることであった．　しかし実際問題として長大なプログラムのどの部分に何を書かねばならないかは　作成者以外は知る由もなく　まして“数値なし”や不条理演算への対応まで考慮すると　第三者の手には負えないのである．

筆者らは初代プログラムの経験から　特注のプログラムテープを作って使用者に手渡すことには反対の意見を持っている．　それはプログラムの種類が多大になることに加えて　一度作成者の手を離れると　そのプログラムはその時点で“進化”をやめ“化石化”して　ついには最新のプログラムとの整合性も失われる結果となる．さらに使用者が無断でプログラムを改造し　それが元で誤動作をした場合　責任が持てないからでもあり　著作権の問題もからんでくる．

そこで使用者が処理成分を算式の形で入力しさえすれ

第1表　データ処理に関係する主要な配列とその役割

用途			変数名	配列規模	説明
使用者からの入力データ	ファイル名	インデックス 共通事項	Idx$	(10)	1．データファイル名　2．作成年月日　3．データ名 4．レコード名　9，成分名ファイル名　10．データタイプ
		インデックス 配列規模	Sc	(5)	1．試料数　2．最大試料数　3．合計可能成分数 4．成分数　5．緯度の項の成分番号(緯経なしなら0)
		成分名 成分名	Nc$	(n)	ノルム計算用プログラムでは，n＝80（データ入力時にノルム鉱物およびD.I.を自動追加）それ以外ではn＝35
		成分名 分子量	W1	(成分数)	データタイプCW(水質用)では分子量÷原子価の値
		データ 試料番号・コード	No$	(試料数，2)	第1列　試料番号　第2列　コード
		データ 化学分析値等	D	(試料数，成分数)	最大試料数は1500または22500÷成分数の商のうち少ない方
データの検索	データのサブコード		Cd$	(4)	No$(I, 2)から割り出された4つのサブコードに対応
	指定コード	コード	C1$	(9，4，2)	組数，サブコード数，{初値/終値} に対応
	指定コード	打点記号	C0$	(9)	指定組数の指示子を兼ねる
データ処理	序列用		N	(試料数)	No$(N(I), 1) No$(N(I), 2) D(N(I), J)の形で用いて、データ処理の順序を定める
	処理用		P	(6，n)	コード別印刷プログラムでは第1－6行を，それ以外では第1行だけ使用。列数はノルム計算用n＝77，それ以外n＝36。コード別印刷プログラムでの追加式の計算用にノルム用では末尾2列，それ以外では1列を使用
	ノルム計算	分析値モル数	M1	(成分数)	
	ノルム計算	ノルム値モル数	M2	(24)	ノルム計算用プログラムだけ ノルム鉱物数に対応
	ノルム計算	ノルム鉱物分子量	W2	(24)	
	成分式の計算	成分式	Cmp$	(11)	1回の処理で指定できる成分式の最大本数に対応
	成分式の計算	式コード	X$	(11)	
	成分式の計算	準備用 成分番号	B1	(11，10)	行数は成分式の最大本数 列数は成分式中の項数 } に対応
	成分式の計算	準備用 定数	B2	(11，10)	同　上
	成分式の計算	項数指示子	Jbt	(11)	B1とB2からQへ数値を移すときに使用
	成分式の計算	演算	Q	(10)	
	成分式の計算	括弧	Pa	(3，5)	行要素は括弧が開いた時点での列数は入れ子の数 1．配列Q用指示子＋1の値 2．演算子コード 3．関数コード
	成分式の計算	演算結果	Plt	(11)	
	成分式の計算	代入先成分番号	Be	(11)	代入先のないとき 0
	雑用		Rd	(1,100)	プログラムごとにREDIM文で配列を変形して使用 数値の出力けた数指示・度数分布図の階級別数値・その他

注：試料数＝Sc(1)　　成分数＝Sc(4)

第2表　成分式に用いる記号

a．演算用の記号

記　号	用　途	式コードでの内容		
		記　号	種　別	値
＋	加　算	左の記号と同じ	演算子	1
－	減　算			－1
*	乗　算			2
/	除　算			－2
(	入れ子			0*
)	同　上		－	－
^	累　乗		－	－
log	常用対数	L	関数	1
ln	自然対数	N		2

*〝(〞は演算子ではないので値を0と定義する。

b．算式に付随する記号

記　号	用　途	例
[　]	上記a.と同じ記号を伴う成分名の指定	[H_2O+]
＝	計算結果の代入先指定	Total=FeO+MgO
;	式の末尾と別名表示文字との区切	T. FeO; Total FeO

ば　それが少々複雑なものであっても計算が出来てしまう方法を考案した（吉井・佐藤，1982）．　これで永年の懸案が一挙に解決した．

b）　基本原理

式の計算機能としては四則混合算・累乗および対数（logまたはln）計算が可能で　式中では括弧が多重に使用できる．　具体例は本誌第347号（吉井・佐藤，1983a）の第1表に示したのでここでは省略する．

計算式に書くことのできる記号を第2表に示す．

演算の基本原理はまことに簡明である．すなわち

ア．すべての演算は加算に帰着する．

イ．式の中で変数と演算子が交互し　原則的には前者が1個多いとき演算が可能である．

（〝原則〞から外れる場合とは，＋または－の記号で始まる算式で，これは変数と演算子とが同数になるが，演算可能である）．

これらについて説明しよう．

まず　つぎのような算式を考える．

$$A+B*C-D\cdots\cdots\cdots\cdots(1)$$

演算の途中では加減算は実行せず乗除算だけを行う．

$$B_1=B*C$$

さらに

$$D_1=(-D)$$

とすれば，最初の式は

$$A+B_1+D_1\cdots\cdots\cdots\cdots(2)$$

となり　演算は必ず合計計算で終る．　つまり上記ア．の意味するところである．

以上がこの方式の肝心な点である．　ここで式(1)の内容を見ると　変数と演算子との関係がイ．の条件を満足している．　これは　このシステムで変数および定数を演算子と分離する際に必要な条件となる．

では　つぎの場合を考える．

$$A+B*C/((D+E)*F-G)\cdots\cdots\cdots\cdots(3)$$

これは典型的な四則混合算で　2重に括弧が用いられている．　この式では当然最も内側の括弧内の計算から順に行うのが算術上の規約である．　人間にはパターン認識という強力な機能が備わっているので　式のどこに入れ子（括弧で囲まれた算式）があるか即座に判断でき計算手順もそこから着手される．　しかし計算機のプログラムとしては　そのような手順によらず　あくまでも式の文字を左から順に解読して処理を進める方が技術的にも容易である．　そこで改めて算式中の括弧の役割について考えてみる必要が生じる．

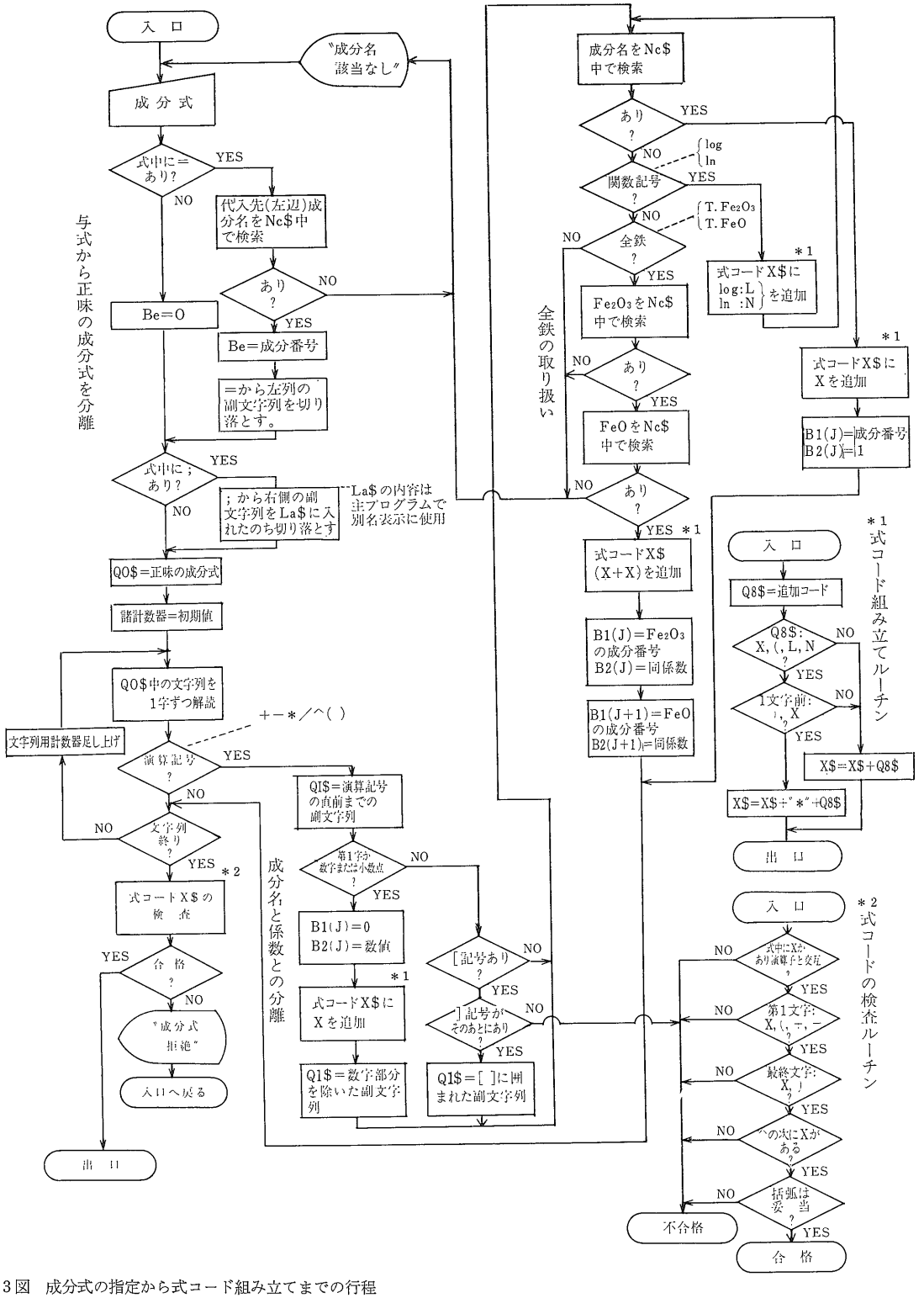
算式を観察すると　最も内側にある括弧から順に閉じて行くことがわかる．　だからその括弧がどの項の直前で開き　さらにはその括弧の直前にどのような演算子および数学関数（ここではlogとln）があるかを記録しておけば　括弧に囲まれた区間内の項およびその区間と　その前項との演算関係を計算機に認識させることができる．これを演算状況の記録と呼ぶことにしよう．

つまり

ウ．〝(〞記号は括弧が開いた時点での演算状況の記録
　　〝)〞記号は入れ子になっている区間の合計計算命令

という機能をもつと考えられる．

では　(3)式の演算を実行しよう．　まず最初の加算は行わず　2番目の演算子(*)に従って乗算を行う．



第3図　成分式の指定から式コード組み立てまでの行程

図の中の変数内容については第1表参照．ただしこの図では成分式を1本分だけ取り扱うことにしてあるので第1表よりも配列規模が一次元分低く示してある．

$$B_1=B*C$$

つぎに　上記ウ．により　括弧の直前が除算記号でありその括弧の直後にある変数(D)が式の第4項であることを記録する（記録方法についてはあとで述べる）．　引き続く2番目の括弧に対しても同じ手続きをする．　ただしその直前に括弧があるが　括弧は演算子ではないのでその旨を記録する．

内側の括弧が閉じた時点で　その括弧が開いた際の記録を引用してこの区間の合計計算を行う．

$$D_1=D+E$$

この段階で(3)式は

$$A+B_1/(D_1*F-G)\cdots\cdots\cdots\cdots(4)$$

と変形されている。　つぎに乗算

$$D_2=D_1*F$$

を行い，次の減算は行わず，最後の括弧閉じの際に，その入れ子区間の合計計算を行う．　すなわち

まず　　$G_1=-G$
続いて　　$D_3=D_2+G_1$

そして外側括弧が開いた際に記録した括弧直前の演算子(／)を用いて

$$B_2=B_1/D_3\quad(\text{ただし}D_3\neq 0\text{とする})$$

最後に残された項の合計

$$\text{Ans.}=A+B_2$$

を実行する．　この値(Ans.)が与式(3)の答である．

3.　演算の準備

成分式の指定行程での具体的な操作は前回省略したので話を戻そう。

使用者が入力した成分式は文字データとして取り扱われ　この文字から成分名・係数・演算記号を判別する．そして　式コードを組み立てる．　与式の演算は式コードの文字をたどりながら実行する（吉井・佐藤，1983a, 1983b)．

では成分式の解読と式コードの組み立てについて述べよう。

なお　これらの行程の冒頭に与式の文字列から　計算結果の代入先を指定する記号＝（等号）と　算式を別名で表示させる文字と算式との区切り記号；（セミコロン）を認識し　これらに挟まれた副文字列を正味の算式として処理する行程がある（吉井・佐藤，1983b, 1984)。　しかし　これらは演算行程とは直接関係ないので　その説明は省略して第3図の中で示すにとどめる．

a)　成分式の解読

使用者が入力した成分式を計算機にどのように読ませて演算させるのか具体例で説明しよう．

いま与えられた式がつぎのようであったとする．

$$100\text{MgO}/(\text{T.FeO}+\text{MgO})$$

この文字列データを算式として認識する行程の流れ図を第3図に示す．　上記のデータは文字列変数Q0$に入れられる．　この中の文字を1字ずつ解読して演算記号

＋　－　＊　／　(　)　＾

を区切りとして式中の項を分離する．　この操作で上の式は取りあえず　100MgO，　T.FeO，　MgO　の3群に分けられる．　つぎに成分名と係数とが分離されT.FeOは全鉄の意味であるから Fe_2O_3 と FeO に分解される．結局計算に係る項は　この場合

$$100,\quad \text{MgO},\quad Fe_2O_3,\quad \text{FeO},\quad \text{MgO}$$

の5項となる．　各成分は成分名ファイル中で存在が確認され　成分番号が配列B1に式の項の順に記入される．定数の場合は成分番号を0とする．　これとは別に配列B2が用意されている．　ここには定数の値または　全鉄の値を算出するのに必要な係数を配列B1と同じ列要素に書き込む．　上記以外の一般の成分に対しては1を入れる．

実際のプログラムでは同時に多数の成分式を指定しその計算が行われる（“成分同士の計算”プログラムでは11本まで)．　しかし　ここでは簡単にするためただ1本の式（例えば“度数分布図”の場合）だけ取り扱うことにしよう．　したがって成分式に関する配列規模は実際（第1表）より一次元だけ低く記す．

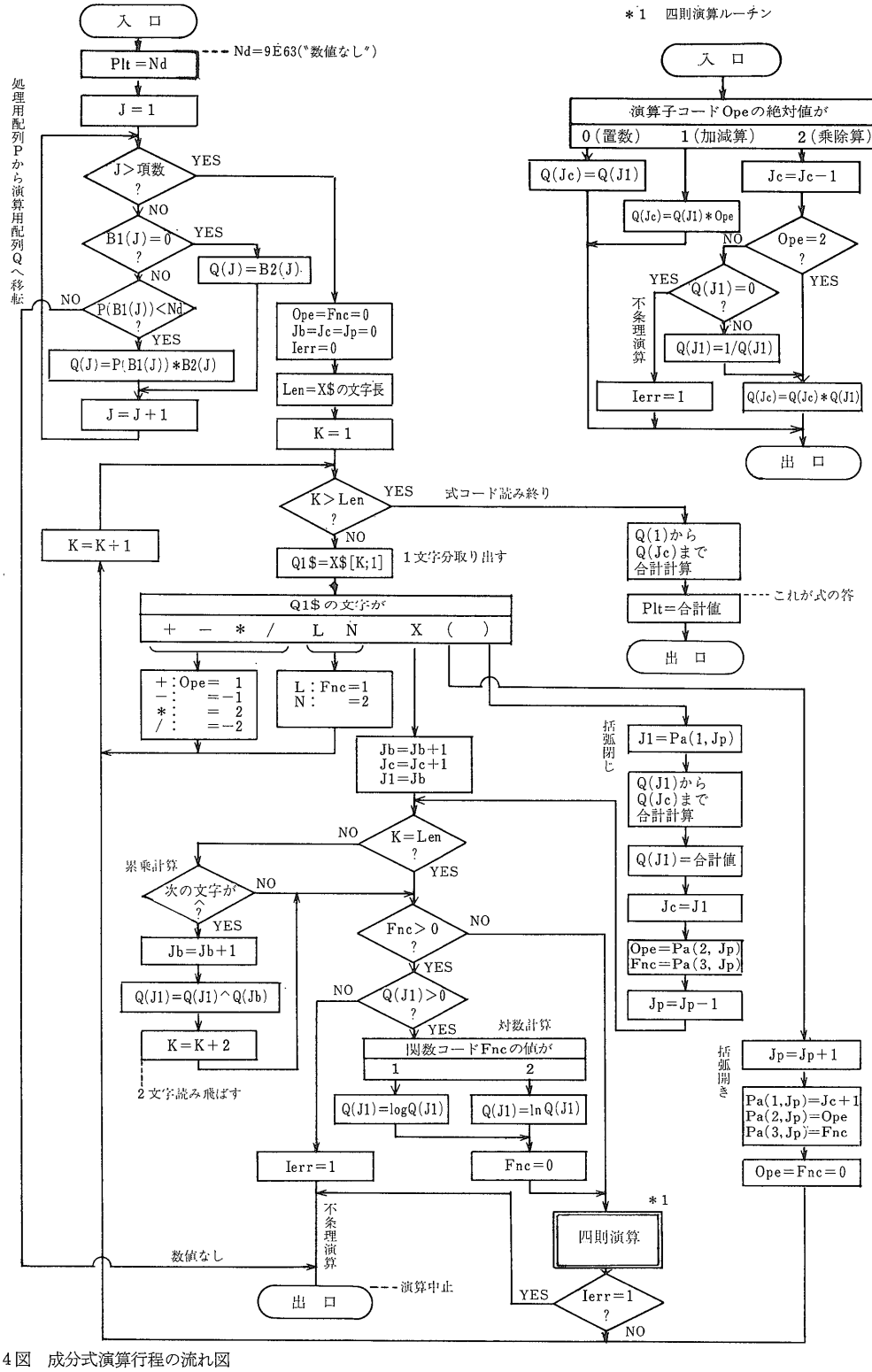
b)　式コードの組み立て

成分式中の成分名と定数をすべてXで表わした式を式コードと呼ぶ（吉井・佐藤，1983b)。

与式の式コードは　つぎのようになる．

$$X*X/((X+X)+X)$$

式コードを組み立てる際には，つぎのような補足を行って与式を演算可能な状態にする。



第4図　成分式演算行程の流れ図

一見複雑に見えるけれども　その主要部は右上の四則演算ルーチンと括弧の開閉に関する部分である.

XX → X＊X
X(→ X＊(
)X →)＊X
)(→)＊(

すなわち式中で乗算記号が省略されている場合は　これを補う仕組である．　したがって 100＊MgO の式は100MgO と書いて差し支えない．　また　全鉄に対しては内容的には2成分分で　しかも両者が一体となっているので括弧が自動的に1組使われて

(X+X)

とされる．

式コードは文字列変数 X$ に記入されて　これで演算の準備が完了する．

4. 演算の実行

演算行程を第4図に示す．　そのあらましはつぎのとおりである．

ア．演算すべき成分の値を配列Pから演算用配列Qへ移す．
イ．文字列変数X$中の式コードに従い演算を実行する．
ウ．最後に配列Qに残った数値の合計をしてその値を変数Pltに与える．

では　主要な事項について　つぎに述べよう．

a) 演算用配列への数値入力

処理用配列Pのどの要素について計算するかは　準備用配列B1とB2に与式の項の順に記憶されている．　また各項の係数は配列B2の対応する列要素に入れられているから演算用配列への数値の入力は　式の第J項に対して

Q(J)＝B2(J)＊P(B1(J))

ただし　配列Pの値に9E63（数値なし）が式の項に 1 つでも含まれていれば計算不能と判断されて　この操作はされず，ルーチンの出口へ直行する．　算式の答を入れる変数 Plt は 9E53 に初期設定されているので　結局この場合は“数値なし”が演算結果となる．

定数項の場合は　配列B1 の列要素は 0 で　B2 にその値が入っているので，つぎの操作を行う．

Q(J)＝B2(J)

b) 演算に関する補助変数

式コードの文字列の解読に先立って　演算に用いるつぎの変数の値をすべて 0 に初期設定する．

Ope：演算子の種類を数値（演算子コードと呼ぶ）として記憶する．
Fnc：数学関数の種類を数値（関数コード）として記憶する．
Jb　：演算用配列Qから数値を取り出す場合の列番号のポインタ（指示子）．
Jc　：配列Qに計算結果を入力する場合の列番号を指示するポインタ．
Jp　：括弧用配列Paへの数値入出力をする場合の列番号のポインタ．
Ierr：不条理演算が行われようとしたときに1の値となり，“異常事態”を伝達するフラグ．

なお　これ以外に補助的に用いられる変数として

K　：式コードX$から文字を取り出すポインタ．
J1　：Jb, Jcの値を仲介するポインタ．

などがある．

c) 式コードの解読に係る変数

X$に入れられている文字を1文字ずつ読んで　その内容に従ってつぎの操作を行う．

その文字が下記のとき演算子コードOpeに値を与える．

＋　：　Ope＝　1　　（加算）
－　：　Ope＝－1　　（減算）
＊　：　Ope＝　2　　（乗算）
／　：　Ope＝－2　　（除算）

つぎの場合は関数コードFncに値を与える．

L　：　Fnc＝1　　（常用対数 log）
N　：　Fnc＝2　　（自然対数 ln）

d) 文字がXであるときの操作

Xの文字が読み取られると　最初にポインタJbおよびJcを1だけ足し上げる．

つぎに　J1＝Jbを行う．

そして　式コードの最終文字でない場合は累乗の記号へがXの直後にあるかどうか判断し　存在すれば　Jbの値をさらに1だけ足し上げる．　への次には必ずXがあるので　それに対応する値を求めて累乗計算を行う．

Q(J1)＝Q(J1)へQ(Jb)　　（ここにJ1＝Jb−1）

このあと　X$用のポインタKをへとXの2文字分足し上げる．　すなわち

```
Components: 100MgO/(T.FeO+MgO)
Formula-code: X*X/((X+X)+X)

  P(7)=  .90      P(4)=  .47      P(5)= 3.06      P(7)=  .90
B(1,1)= 0       B(1,2)= 7       B(1,3)= 4       B(1,4)= 5       B(1,5)= 7
B(2,1)=100.00   B(2,2)=  1.00   B(2,3)=   .90   B(2,4)=  1.00   B(2,5)=  1.00
-------------------------------------------------------------------------
Step= 0  Symbol:       Pointer: Jb= 0  Jc= 0  Jp= 0     Code: Ope= 0
Q(1)=100.000  Q(2)=    .900  Q(3)=    .423  Q(4)=  3.060  Q(5)=    .900
-------------------------------------------------------------------------
Step= 1  Symbol: X     Pointer: Jb= 1  Jc= 1  Jp= 0     Code: Ope= 0
Q(1)=100.000  Q(2)=    .900  Q(3)=    .423  Q(4)=  3.060  Q(5)=    .900
-------------------------------------------------------------------------
Step= 2  Symbol: *     Pointer: Jb= 1  Jc= 1  Jp= 0     Code: Ope= 2
Q(1)=100.000  Q(2)=    .900  Q(3)=    .423  Q(4)=  3.060  Q(5)=    .900
-------------------------------------------------------------------------
Step= 3  Symbol: X     Pointer: Jb= 2  Jc= 1  Jp= 0     Code: Ope= 2
Q(1)= 90.000                  Q(3)=    .423  Q(4)=  3.060  Q(5)=    .900
-------------------------------------------------------------------------
Step= 4  Symbol: /     Pointer: Jb= 2  Jc= 1  Jp= 0     Code: Ope=-2
Q(1)= 90.000                  Q(3)=    .423  Q(4)=  3.060  Q(5)=    .900
-------------------------------------------------------------------------
Step= 5  Symbol: (     Pointer: Jb= 2  Jc= 1  Jp= 1     Code: Ope= 0
Q(1)= 90.000                  Q(3)=    .423  Q(4)=  3.060  Q(5)=    .900
Pa(1,1)= 2
Pa(2,1)=-2
Pa(3,1)= 0
-------------------------------------------------------------------------
Step= 6  Symbol: (     Pointer: Jb= 2  Jc= 1  Jp= 2     Code: Ope= 0
Q(1)= 90.000                  Q(3)=    .423  Q(4)=  3.060  Q(5)=    .900
Pa(1,1)= 2    Pa(1,2)= 2
Pa(2,1)=-2    Pa(2,2)= 0
Pa(3,1)= 0    Pa(3,2)= 0
-------------------------------------------------------------------------
Step= 7  Symbol: X     Pointer: Jb= 3  Jc= 2  Jp= 2     Code: Ope= 0
Q(1)= 90.000  Q(2)=    .423                  Q(4)=  3.060  Q(5)=    .900
Pa(1,1)= 2    Pa(1,2)= 2
Pa(2,1)=-2    Pa(2,2)= 0
Pa(3,1)= 0    Pa(3,2)= 0
-------------------------------------------------------------------------
Step= 8  Symbol: +     Pointer: Jb= 3  Jc= 2  Jp= 2     Code: Ope= 1
Q(1)= 90.000  Q(2)=    .423                  Q(4)=  3.060  Q(5)=    .900
Pa(1,1)= 2    Pa(1,2)= 2
Pa(2,1)=-2    Pa(2,2)= 0
Pa(3,1)= 0    Pa(3,2)= 0
-------------------------------------------------------------------------
Step= 9  Symbol: X     Pointer: Jb= 4  Jc= 3  Jp= 2     Code: Ope= 1
Q(1)= 90.000  Q(2)=    .423  Q(3)=  3.060.                   Q(5)=    .900
Pa(1,1)= 2    Pa(1,2)= 2
Pa(2,1)=-2    Pa(2,2)= 0
Pa(3,1)= 0    Pa(3,2)= 0
-------------------------------------------------------------------------
Step=10  Symbol: )     Pointer: Jb= 4  Jc= 2  Jp= 1     Code: Ope= 0
Q(1)= 90.000  Q(2)=  3.483                              Q(5)=    .900
Pa(1,1)= 2
Pa(2,1)=-2
Pa(3,1)= 0
-------------------------------------------------------------------------
Step=11  Symbol: +     Pointer: Jb= 4  Jc= 2  Jp= 1     Code: Ope= 1
Q(1)= 90.000  Q(2)=  3.483                              Q(5)=    .900
Pa(1,1)= 2
Pa(2,1)=-2
Pa(3,1)= 0
-------------------------------------------------------------------------
Step=12  Symbol: X     Pointer: Jb= 5  Jc= 3  Jp= 1     Code: Ope= 1
Q(1)= 90.000  Q(2)=  3.483  Q(3)=    .900
Pa(1,1)= 2
Pa(2,1)=-2
Pa(3,1)= 0
-------------------------------------------------------------------------
Step=13  Symbol: )     Pointer: Jb= 5  Jc= 1  Jp= 0     Code: Ope=-2
Q(1)= 20.534
-------------------------------------------------------------------------
Result=20.534
```

第5図　成分式演算実行時の各変数内容の変化

各行程の最終結果を示す．数値移動後の配列Qの内容は図を見易くするため変数名ごと抹消してある．また対数関数も不要なので変数 Fnc の表示も省略した．

K=K+2

累乗計算の場合はXの次の字の判断を要するので　手続きは面倒だけれども　対数計算ではXに先行する関数コードが正のとき　その値に従ってそのときのポインタJ1が示す配列Qの値に対して　たとえば常用対数なら

Q(J1)=logQ(J1)　　（ここにJ1=Jb）

を行えばよい．　この計算のあとの関数コードの値を

Fnc=0

とするのがポイントである．

上記の計算のあとで四則演算に移る．　操作は演算子コード Ope の値に従う．

ア．Ope＝0 とき　置数すなわち珠算での“御破算”のあとに数値を置く操作である．　これは式コードの第1字がXであるか　または開き括弧の直後にXが続く場合で　操作は配列内での数値移動である．

$$Q(Jc)=Q(J1)$$

イ．Ope＝±1のときは加減算で　同じく配列内の数値移動操作を行う．　ただし Ope の符号が加算と減算とを自動的に区別させる．　すなわち

$$Q(Jc)=Q(J1)*Ope$$

ウ．Ope＝±2 の場合は配列Q内の要素同士の乗除算を実行する．　手順はまず　Jcの値を1だけ減らす．　Ope＝2のときは　つぎの乗算を実行する．

$$Q(Jc)=Q(Jc)*Q(J1)$$

Ope＝−2では　Q(J1) の値が0でないとき

$$Q(J1)=1/Q(J1)$$

を実行したのちに乗算の操作と合流する．

e)　文字が開き括弧のとき

括弧用二次元配列Pa 用のポインタ Jp を1だけ足し上げ，つぎの値を Pa に記入する．

$$Pa(1,Jp)=Jc+1$$
$$Pa(2,Jp)=Ope$$
$$Pa(3,Jp)=Fnc$$

以上は括弧が閉じた際の演算の準備である．そして

$$Ope=Fnc=0$$

とする．

f)　閉じ括弧のとき

まず　配列 Pa の内容を基に入れ子の区間の合計計算を行う．　すなわちポインタの初値を

$$J1=Pa(1,Jp)$$

によって求め　合計値をつぎのようにQ(J1) へ代入する．

$$Q(J1)=Q(J1)+Q(J1+1)+\cdots+Q(Jc)$$

そして　　$Jc=J1$

さらに　　$Ope=Pa(2,Jp)$
　　$Fnc=Pa(3,Jp)$

の操作で　開き括弧の直前にあった演算子コードと関数コードを呼び戻す．　括弧が1つはずれたので　括弧用ポインタPaの値を1だけ減らす．

このあと上に述べた累乗・対数・四則計算のルーチンへ入って　入れ子になっていた区間の値とその前の項との演算を行う．

g)　文字コードを読み終ったとき

配列Qの第1列から第Jc列までの和を求め変数 Plt に代入する．　すなわち

$$Plt=Q(1)+Q(2)+\cdots+Q(Jc)$$

この Plt の値が与式の演算結果である．

h)　演算実行の実例

具体例として，

Fe_2O_3	0.47%
FeO	3.06
MgO	0.90

のとき　100MgO/(T.FeO＋MgO)

の値を重量百分率で求める場合の各行程での変数内容の変化を第5図に示す．　なお　括弧用変数 Pa の内容は括弧が開いている区間だけを示す．　この例では最後が除算で終るので　Q(1)の値そのものが演算結果となる．

5.　方式の特徴

文字データを式コードに変換して演算を実行する方式は　上記の長い説明と第4図を見た限りでは　大変複雑な手順を踏んでいるように感じる向きもあろうが　第4図の右上に描いた部分が演算の主要部で　カラクリはきわめて単純である．　取り扱う式の項数や括弧の重ね数も配列規模の問題であり　演算ルーチンのプログラムとは係りない．

演算式も使用者の便を計り　筆算での表記法に極力近付けてあり　乗算記号を省略しても演算ができるので

$3.33a$，　$2(a+b)(c+d)$　などと書いてよい．

計算の優先順位も筆算での規約を当然のことながら遵守しており　$\log x^2$ を計算するには　logx^2 と入力すれば　$2\log x$ と同じ結果を得る．　ちなみに　一般のコンピュータでは上記のように入力すると，

$$(\log x)^2$$

を実行する．　この例では　なまじコンピュータの知識のある人が操作をすると誤算となり　皮肉な結果となる．

演算の仕組そのものが簡明なので　与式が簡単なら迅速に計算が終了し　その一方で込み入った算式も　それなりにこなせ　将来への拡張性にも富んでいる．

誤動作の心配はまずないが　成分式とともに式コードも印刷されるので　与式が正しく解釈されているかどうかも確認できるように配慮がなされている．

6. あとがき

演算方式の研究は1981年12月から正月休みにかけての時期にさまざまなモデルを考案しながら行った．　机上での紙と鉛筆だけの検討の結果　上記の方式に決定した．

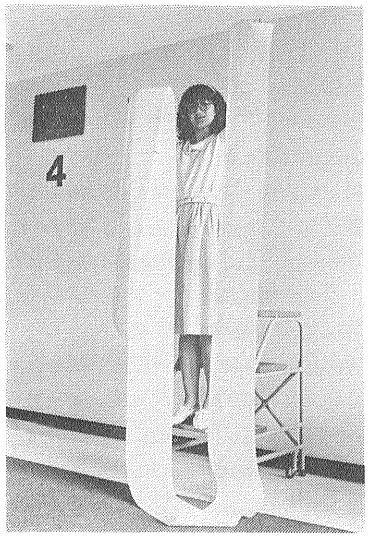
この案を基に BASIC による実験プログラムを組み　幸にも主要部はわずか半日で通った．　演算ルーチンのプログラムミスは　直ちに誤算につながる恐ろしい事態となるので　テストランは殊更過酷に執ようなまで行った上で　データ処理用のすべてのプログラムに実装した．

成分式の演算法を部内で公開した当初は“計算機が計算するのは当り前で　少しも目新しくない”．とか“算式はプログラムに直接書けば済むことで　そんな事が地球化学に貢献するとは思えない”．　と言った批判も浴びたけれども　このような理論派からの批判とは裏腹に　実際にキーボードに向った使用者からは例外なく好評を得た．　そしてこの時点を境に筆者らのシステムの利用者が部の内外を問わず急増し　最近は GEOCAPS に対する問い合わせが所外や在外邦人からも寄せられ　見学にお出ましの方々もあった．　そして　GEOCAPSおよびその前身のシステムを利用した研究論文が順次公表されて来ている．　これは筆者らの大きな喜びでもある．

もちろん筆者らも一使用者として　このシステムの機能を活用し　新たな観点から従来の地球化学的データの解析を進めている．

最後に　このプログラムシステムの規模について付け加えておこう．　プログラムセグメントの数は目下のところ約60である．　BASIC としての行数は延べ7200行を超えている．　GEOCAPS では　これらの中から必要なセグメントだけ連結し実行するので　これら全部が計算機に入ることはない．　計算機に入力される最長のプログラムは　ノルム計算付き積層型 X-Y 相関図で約1500行に及ぶ．　プログラムリストは　つねにセグメント単位で保存してあるが　試みに　この最長のプログラムをプリントしたら　優に５m以上に達した（第６図）．

プログラムは文章と同様に冗長であってはならない．しかし　使い勝手を良くするためには　懇切な説明書と同じくどうしても行数がふえてしまう．　いかに簡明で使い易いプログラムとするかが，作る人の腕の見せどころである．



第６図　GEOCAPS 中最長のプログラムリスト

プログラムは長いばかりが能ではない．だが使い易くするためにどうしてもこの程度にはなる．これはノルム計算付積層型 X-Y 相関図で約1500行ある．ちなみにこのお嬢さんの身長は167cm．

引　用　文　献

吉井守正（1980）文字列を使ったコードによるデータの選択．地質ニュース，No. 315，p. 13-17.

———（1983）　GEOCAPSでのデータの入力とソート．地質ニュース，No. 349，p. 58-63.

———・佐藤岱生（1981）岩石化学データ処理システムのあらまし．地質ニュース，No. 321，p. 40-45.

———・———（1982）BASICによる算法入力ソフトウエア．　日本鉱山地質学会・日本岩石鉱物鉱床学会（“三鉱学会”）昭和57年秋期連合学術講演会講演要旨集，p. 51.

———・———（1983a）改名GEOCAPS．　地質ニュース，No. 347，p. 57-64.

———・———（1983b）BASIC による地球化学データ解析システム GEOCAPS のあらまし．情報地質，No. 8，p. 21-40.

———・———（1984）GEOCAPS でのデータ処理手順（１）　——処理条件の設定．地質ニュース，No. 356，p. 40—48.