C 5. 数式処理システムについて

大野瑞夫（慶応義塾大学）

1 KFMSの目的

数式処理システムを設計するにあたり，その目的としたことは次のようなことである．

① 数式（FORMULA）の微分，積分，展開，式のまとめ等の数式処理（FORMULA MANIPULATION）を人間にかわり，電子計算機を使つて行う．

② 数式処理のシステムは，専門家でなくても，容易に，かつ実用的に利用できる形とする．

③ 数式処理だけではなく，一般に必要とされる程度の数値計算も行うことのできるシステムとする．

2 数式処理システムの問題点とKFMS

数式処理システムといつても，その対象とする数式，処理能力，処理方法などについて多くの問題がある．以下にその問題点およびそれに対するKFMSの考え方を説明する．

① 数式処理システムの設計

a．汎目的，特殊目的

各種の数式処理システムが作られているが，それらは，大きく2種類に分けることができる．すなわち，数式処理なら一応なんでもできるというシステムと，微分なら微分だけ，積分なら積分だけという，特定の数式処理しかできないものとである．

これの選択は，もちろん使用目的によるわけであるが，KFMSは，タイムシエアリングシステムの一部として使う計画から，できるだけ広範な数式処理を行えるシステムを計画した．

汎目的システムの長所として考えられるのは，たとえば，数式の単純化を行うルーチンのように，どの種類の処理をするにも必要とされるものがあるため，そのようなルーチンを個々に作る必要のある特殊目的のシステムと比べ，システム全体の効率が高くなるということがいえる．また，より広い範囲の人々が，種々の目的のために数式処理システムを利用することが期待できる．

他方，短所としては，何でも処理できるかわりに，個々の問題，特に，多項式の処理などの場合，特殊目的のシステムと比べて，能率の悪いシステムになる可能性があることが考えられる．

b．言　語

KFMSは，数式処理を行うために設計した単純な言語を使う．

言語の柔軟性という観点から見ると，目的としてかかげた，使い易いシステムということは，柔軟性を欠く原因ともなる．すなわち，ある目的のために，便利に作つてある言語はその目的以外には，全く不便な言語となるからである．このような問題を解決するにはFORMULA ALGOLのように，利用者にプログラムを書かせる形にするか，システムが利用者によつて教育されうるように設計されている必要がある．KFMSでは，現在そこまで考えていない．

c．バッチ処理形式，人間機械会話形式

容易にかつ，実用的に利用できる数式処理システムとしては，研究室のタイプライターを打てば，すぐ答の返つてくるシステムの方が良いことはいうまでもないので，人間機械会話形式を採用した．また，KFMSは，数式処理の卓上計算機版とした．そのため，1つの命令に対する答を利用者が確認しながら次々に仕事を進めてゆくこともできる．

この方法の利点は，数式処理では式の識別など，まだ計算機では十分でない面を人間に補助してもらうことができること．及び，数値計算とは比較にならない多くの情報を持つた結果が生ずるので，次にどのステップに進むかの判断を利用者に委ねることができる．ということがあげられる．

数値計算の場合，途中結果の値により，いろいろ判断ができるが，数式処理の途中結果は数値だけではなく，たとえば，ある変数が有理式の分母にあるか，とか，数式中に三角函数があるかとか，非常に多岐にわたる．この種の判断をストアドプログラムで持つということは問題が多いと思われる．

一方，欠点としては，ループになつた計算，たとえば，FORTRANのDOループなどのようなものが，行えないことがあげられる．

d．入出力の形状

入力としては，誰でも容易に使えるという目的から，一般に使われている数式をそのまま受けつけられることが理想ではあるが，普通，2次元入力はできないため，FORTRANの記法程度が限度となる．

また，何行にもわたる多くの括弧のついた式は，非常に読みづらいのでなんとかする必要があるが，KFMSでは，使用できる装置がタイプライターに限られているので，何ら解決は図られていない．

② 方程式，式

数式処理における方程式(EQUATION)，式(EXPRESSION)の区別をはつきりさせておく．

式は，代数などにおける，Y，A+B，X^2，$(C-D)^2$，SIN(X) のようなもので，方程式は式を2組等号で結んだものとする．たとえば，X＝A＋B，$E=MC^2$，X＋Y＝A／B のようなものであるとする．

以上の定義によると，数式処理は方程式，あるいは，式をいろいろ操作することである．たとえば，$X=(A+B)^2$ を $X=A^2+2AB+B^2$ と展開したり，$A+B=(C+B)^2/(C+B)$ を A＝C と変型したりすることである．

ここで，FORTRAN の算術ステートメントのように方程式の左辺に式を自由に書くことのできない方程式が問題となる．この種の方程式の等号は前例の方程式の等号とは意味が異るので，この種の方程式は以後，方程式とはいわず，式の一種と考える．

電子計算機で数式処理を行うという場合，上の定義での方程式を処理するか，式の処理だけをするか，は大きな問題である．たとえば，IBMのFORMAC は，方程式の処理は行わず，式の処理だけである．方程式の処理も行つているのは，MACのMATHLABぐらいである．

KFMSでは，式の処理だけを行う．この文中で，KFMSに関して数式という言葉が使われるときは，式のことであり，方程式を意味していない．

③ 算術演算

数式処理システムで，一般に必要とされる数値計算を行う という場合整数，実数の四則演演算の他に，分数計算が多くの場合非常に重要になる．たとえば，(3／8)X+4／8 という式を，0.333X+0.5 とするよりも，(1／3)X+1／2 とする方が実用的である．特に，$\sum x^N/N!$ のような式の場合はいうまでもない．

この分数計算には，分数の和算における通分，約分，等の一般の数値計算では関係なかつたわずらわしい計算が入つてくる．

KFMSでは，整数，実数，分数の四則演算を行う．その際，式中に整数と実数が混つていれば，それはすべて実数とみなし，実数の計算をしてしまう．また，分数が，分子，分母共に整数なら分数として取扱うし，実数が1つでもあれば除算を行い，実数にする．たとえば，

X＝5.5＋3　　結果　X＝8.5

$Y=\frac{3}{5}+\frac{2}{10}$　　結果　$Y=\frac{4}{5}$

$Z=\frac{1}{2}+0.5$　　結果　Z＝1.0

他に，倍精度計算，複素数計算も考えられるが，KFMSでは考えていない．

④ 単 純 化

この問題は，数式処理において式の識別と共に，最も重要なものであると考えられる．すなわち，どのような数式処理を行つても，それをある「より単純」であると考えられる形になおすことがどうしても必要となるからである．

たとえば，Y＝(X＋1)(X＋2)^{2}を，微分の公式$(uv)'=u'v+uv'$で微分すると，Y'＝1・(X+2)^2＋(X+1)・2・(X+2)・1となる．これを，Y'＝(X+2)^2＋2・(X+1)(X+2)とすることは，一般にどうしても必要であり，当然使用者が指示しなくても行つて欲しい単純化である．

一方，$x(x+2)+1$の「より単純」な形というのは，$(x+1)^2$であるか，または，x^2+2x+1か，そのままの式であるかというのは，場合によつて異るため，なんとも言うことができない．このなにが「単純」であるかということがはつきりしないという問題については，使用者に式を展開するかどうかなど，指示させることにより解決を図つている．

もう1つ，数式を正規化するという意味で，単純化を行う場合がある．これは，2つの式が同じものかどうかを容易に見わけるため，あるいは，判別する数式の種類を少くして，システムの設計を楽にするために行われる．たとえば，FORMACでは，B+A+(D*C) → A+B+(C*D)，(A* B)**C → A**(B*C)のような正規化が行われる．

KFMSの場合，システム自動的に行う単純化は，表1の四種類である．正規化については，現在，余分な括弧を除くことだけを行つている．

例　A*((B+C))　　A*(B+C)

表1　自動的に行う単純化

i)　0，1に関するもの

0＋A	→ A	
0 * A	→ 0	
0 * *A	→ 0	A＞0
A * *0	→ 1	A≠0
0／A	→ 0	A≠0
1 * *A	→ 1	
A * *1	→ A	
1 *A	→ A	
A／1	→ A	

ii)　整数，実数の四則演算およびライブラリー関数のオペランドが，数字だけの場合，計算を実行する．

例　5＋3　→8

2＊5.5　→11.

SIN(0)　→0.

iii)　分数の四則演算，約分

例　8/10　→4/5

1/2+1/3　→5/6

iv)　EXP(LOG(A))→A　　A≠0

LOG(EXP(A))→A　　A≠0

⑤　式の識別

数式処理において，式を識別するというのは，単純化，微分，積分などにおいて非常に重要な問題である．数式について，そのパタンを問題とするとき，数式の内部表示が問題となるが，LISP.で作つたシステムやFORM.ACでは，オペレーターを前に置いたリスト構造になつている．ところが，そのような内部表示は，データを伸縮させたり，数値を計算するには便利だが，高度の式の識別のためにはあまり良いとはいえない．

一般に，数式処理では，繰り返してある規則を適用することにより，できるだけ単純な判断で式の識別を行う方法がとられている．たとえば，KFMSでは，微分を行う場合，ペレーターの種類と微分する変数があるかどうかだけでどの規則を適用するか判断し，それ以上は調べていない．

⑥　数式処理の規則

数値計算を行う場合は，四則演算さえできれば良かつたが，数式処理を行う場合は，多くの，個々の事例に対する規則が必要となる．たとえば，$(A+B)^2 \rightarrow A^2+2AB+B^2$ と展開するようなとき，二項係数を何らかの形で展開の規則に組み込む必要があり，一般的な形の規則を作ることはめんどうである．この場合についていえば，複雑にはなるだろうが，ある規則を作ることが可能であるし，その規則に従えば，必ず答が得られるようになるだろう．

しかし，積分のように，ある一つの規則に従えば，必ず答が得られるということもなく，また，答があるかどうかもわからないものはどう処理したら良いだろうか．たとえば，$\int x^2 \sin(x)dx$ という問題に対し，部分積分を行い，$\int \sin^2(x)dx$ に対しては $\sin^2(x)$ を $(1-\cos 2x)/2$ とおきかえれば良いなどということを，どのように規則として作るかは困難なことである．これには，上に述べた数式の展開や微分のような数式処理とは異る性質の規則が必要であると考えられる．KFMSでは，繰り返して適当な規則をあてはめるという手法を試行錯誤で適用する．

その方法をKFMSで積分を行う場合について説明する．システムには，基本的な公式が教えてあつて，システムに与えられた問題の式は，まず，公式が適用できるかどうか調べられる．公式が適用できれば，それで終りであるが，そうでない場合，たとえば，$\int x(x+1)\,dx$のような積分変数が積の形で2つ以上現われているときは，すべて部分積分．$\int \sin(x+1)\,dx$ のような問題は，置換積分と選り分けられる．部分積分を行う場合，どの項を積分する方の項にするかわからないので，とりあえず，どちらかを適当に選んで積分する．その際，その積分も公式で簡単に解けない場合には，その部分について，再び部分積分なり，置換積分を行う．上の例では，どちらの項を選んでも公式で解くことができる．そうして，何回かそのような試行を行つてうまくゆかない場合には，他方の項について，積分を行う部分積分を実行する．そして，再び，同じ手順を繰り返すことになる．それでもうまくゆかなければ，このシステムでは解くことができない．置換積分の場合は，どの項をどう置き換えれば良いかは，全く試行錯誤に頼る他はないため，いろいろな置換をはじから試すことになる．

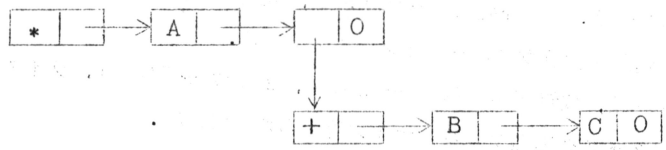
⑦ システム製作上の問題

a．数式の内部表示

数式は，プログラム実行時における変型が容易で，かつ，種々の式の識別も容易な形で貯えられることが望ましい．

LISP，IPL－Vで作られたシステムでは，オペレーター前置の2進木のリスト構造が使われているが，KFMSでも，同じ構造を採用している．

例　A＊(B＋C)



b．記憶装置の容量

数式処理では，パタンを見つける部分のプログラムが非常に大きくなるので，処理能力を拡げると，システムも相当大きくなつてしまう．また，長い数式を取扱うと，データの占める容積も大変なものになるので，システムのオーバーレイ，データの補助記憶装置への格納，引き出し，が必要となつてくる．

この問題は，タイムシエアリングシステムで数式処理を行う場合，記憶装置，補助記憶装置に余裕が少いので無視できない．

c．アツセンブラー，コンパイラー

数式処理システムを作る場合，それを，LISP，IPL－Vのようなリスト処理言語で

作るか，あるいは，アッセンブリ言語で作るか決定しなければならない．

KFMSの場合，リスト処理言語がまだできていないということ，および，使用できる記憶容量が小さいという点により，システムはすべてアッセンブリ言語で書かれている．

KFMS の概説

KFMSは，汎目的数式処理システムであり，タイムシエアリング・システムの一部として，各種の数式処理をタイムシエアリング・ベースで行うことができる．

① システムの構造

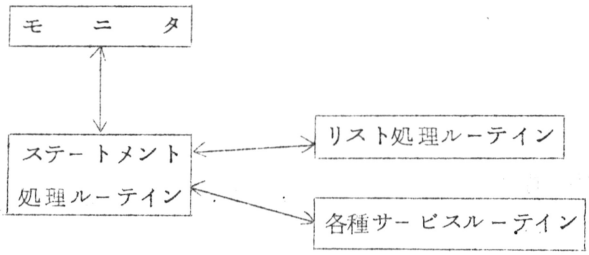


図　1

KFMSは，図1のように大きく分けて考えることができる．まずモニタは，利用者の入力ステートメントを解読し，対応する処理ルーテインにコントロールを渡す．次に，ステートメント処理ルーテインは，積分，微分，数式の外部表示 → 内部表示変換，内部表示 → 外部表示変換，単純化，置換，展開，係数抜取，評価の各ルーテインからなつていて，各ステートメントの処理を行う．リスト処理ルーテインは，数式の内部表示がリスト構造になつているため，その処理を行うサブルーテインの集りである．

各種サービスルーテインは，数値変換（BCD → 2進数，2進数 → BCD）．ライブラリー関数評価，および，スタック処理サブルーテイン等からなつている．

② ステートメントの説明

ここで使う「変数」とは，英字で始まる8文字以内の英数字であつて，「原子変数」とは，原子変数宣言のリストにある変数のことで，式の右辺にだけ書くことができる．また，「式変数」とは，原子変数以外の変数である．以後，説明のため，変数を v で表す．

a．宣言ステートメント

i) 原子変数宣言

ＡＴＯＭＩＣ　v_1 , v_2 , ……, v_j

変数 v_1 , v_2 , ……, v_j を原子変数として登録する.

例　ATOMIC I,J,KEIŌ

b．実行ステートメント

i) 式 定 義

v = 式

任意の式を v と定義する．任意の式は対応する内部表示に変換され，v は，式変数の表に登録される．

このステートメントを書く際，注意しなければならないことは，原子変数と式変数との区別をはつきりさせておくということである．たとえば，次の例のような書き方をすると，同じXという変数が，原子変数と，式変数と2つの表に登録されてしまうことになり，Y＝X＋JのXがどちらだか分らなくなる．この場合，Y＝X＋JのXは，原子変数と見なされる．

例　ATŌMIC . X , I , J
X=J*(SIN(Y))
Y=X+J

ii) 積　分

v_i =INTEG(v_j , v_k)

式 v_j を v_k で積分して，その結果の式を v_i とする．

例　X=J*(SIN(Y))
XI=INTEG(X,Y)
結果　XI=-(J*(COS(Y)))

積分を行う手順として

㋑ 積分を行う式を単純化し，正規化する．

㋺ 積分公式(表2)が適用できるかどうか判断する．
適用できれば，ただちに答が得られる．

㋩ 公式を適用できない式は，部分積分，あるいは置換積分を行う．

の3ステップがある．㋩では，微分ルーテイン，および，積分の3ステップ，㋑，㋺，㋩を使用するが，recursiveな方法は使わない．

表 2　積 分 公 式

1　$\int (f(x)\pm g(x))dx \longrightarrow \int f(x)dx \pm \int g(x)dx$

2　$\int k\,dx \longrightarrow kx$

3　$\int kx^n dx \longrightarrow kx^{n+1}/(n+1)$

4　$\int k/x\,dx \longrightarrow k\ \log|x|$

5 $\int \sin kx\,dx \longrightarrow -\cos kx/k$

6 $\int \cos kx\,dx \longrightarrow \sin kx/k$

7 $\int e^{kx}\,dx \longrightarrow e^{kx}/k$

8 $\int k^x\,dx \longrightarrow k^x/\log k$

9 $\int \sec^2 x\,dx \longrightarrow \tan x$

10 $\int \mathrm{cosec}^2 x\,dx \longrightarrow -\cot x$

11 $\int dx/\sqrt{k^2-x^2} \longrightarrow \sin^{-1}(x/k)$

12 $\int dx/\sqrt{x^2+k} \longrightarrow \log|x+\sqrt{x^2+k}|$

13 $\int dx/(x^2-k^2) \longrightarrow (1/2k)\log|(x-k)/(x+k)|$

14 $\int dx/(x^2+k^2) \longrightarrow (1/k)\tan^{-1}(x/k)$

15 $\int \sqrt{k^2-x^2}\,dx \longrightarrow (1/2)(x\sqrt{k^2-x^2}+k^2\sin^{-1}(x/a))$

16 $\int \sqrt{x^2\pm k^2}\,dx \longrightarrow (1/2)(x\sqrt{x^2\pm k^2}\pm k^2\log|x+\sqrt{x^2\pm k^2}|)$

iii) 微　分

$v_i = \mathrm{DIF}(v_j, v_k)$

式v_jをv_kで微分して，その結果の式をv_iとする．

例．　X=K*(SIN(Y))

　　XD=DIF(X,Y)

　　結果　XD=K*(COS(Y))

微分を行う手順

㋑　微分を行う式を単純化し，正規化する．

㋺　微分公式（表3）を適用する．その際，更に微分する必要のない項（ダッシュのついていない項）には，微分が済んだことを示す旗を立てておき，公式を適用した結果の式に対し，再び微分公式を適用するということを繰り返す．すべての項に微分が終了したことを示す旗が立てば，微分は終了である．

積分の規則とは異り，微分り規則は，同じ規則の繰り返しであり，recursive であるともいえる．

表　3　微　分　公　式

1 $(k)' \longrightarrow 0$

2 $(x)' \longrightarrow 1$

3 $(\Gamma\pm\Delta)' \longrightarrow \Gamma'\pm\Delta'$

4 $(\Gamma\Delta)' \longrightarrow \Gamma'\Delta+\Gamma\Delta'$

5 $(\Gamma/\Delta)' \longrightarrow (\Gamma'\Delta-\Gamma\Delta')/\Delta^2$

6 $(\Gamma^{\Delta})' \longrightarrow \Gamma^{\Delta}(\Delta\Gamma'/\Gamma+\Delta'\log\Gamma)$

7 $(\log\Gamma)' \longrightarrow \Gamma'/\Gamma$

8 $(e^{\Gamma})' \longrightarrow e^{\Gamma}\Gamma'$

9 $(\sin\Gamma)' \longrightarrow \Gamma'\cos\Gamma$

10 $(\cos\Gamma)' \longrightarrow -\Gamma'\sin\Gamma$

11 $(\tan\Gamma)' \longrightarrow \Gamma'\sec^2\Gamma$

12 $(\cot\Gamma)' \longrightarrow -\Gamma\,\mathrm{cosec}\,\Gamma$

13 $(\sec\Gamma)' \longrightarrow \Gamma'\tan\Gamma\sec\Gamma$

14 $(\mathrm{cosec}\,\Gamma)' \longrightarrow -\Gamma'\,\mathrm{cosec}\,\Gamma\cot\Gamma$

15 $(\sin^{-1}\Gamma)' \longrightarrow \Gamma'/\sqrt{1-\Gamma^2}$

16 $(\cos^{-1}\Gamma)' \longrightarrow -\Gamma'/\sqrt{1-\Gamma^2}$

17 $(\tan^{-1}\Gamma)' \longrightarrow \Gamma'-(1+\Gamma^2)$

iii) 評　価

v_i =EVAL v_j,(v_k,e_k),(v_e,c_e),……

式 v_j の変数 v_k に定数(実数，整数，分数のいずれか) c_k，v_e に c_e，…… を代入して，その結果を計算して，v_i とする．

例．　XX=K+(COS(Y))

XE=EVAL XX (K,5),(Y,0)

結果　XE= 6.0

式中に，整数，実数が混つている場合は，整数を実数に変換して計算を行い，結果は実数となる．また，分数計算は，分母，分子共に整数のときだけ行うのであつて，分子か分母に実数があれば，除算を実行して実数にしてしまう．

上の例のように，三角函数の引数に整数を用いた場合は，その整数を実数に変換して，函数を評価している．

算術演算における定数の型の変化

	整　数	実　数	分　数
整　数	整　数	実　数	分　数
実　数	実　数	実　数	実　数
分　数	分　数	実　数	分　数

iv) 置　換

v_i =SUBST v_j,(v_k,v_k'),(v_e,v_e'),……

式 v_j の変数 v_k を v_k'，v_e を v_e' で置き換えてその結果の式を v_i とする．

v_m=SUBST v_i,(v_k',),(v_e',)

式v_iの変数$v_{k'}$,$v_{e'}$が式変数であつて，各々をその式で置き換えた式をv_mとする．

v_i=SUBST v_j

式v_jのすべての式変数を，原子変数だけの式で置き換える．

例．　ATOMIC　A,B,C,D,E

X=A+B

Y=C+D

Z=C+E

XS=SUBST Y,(C,X),(D,Z)

結果　XS=X+Z

XXS=SUBST XS,(X,)

結果　XXS=A+B+Z

XA=SUBST XS

結果　XA=A+B+C+E

vi) 展　開

v_i=EXPAND v_j,(v_k,v_e,……)

式v_jのv_k,v_e,……について，順に展開する．

例．　A=((X+B)**2)+((B+C)*D)

AAE=EXPAND A,(X,B)

結果　AAE=X^2+2XB+B^2+BD+CD

vii) 係数抜取り

v_i=COEF(v_j,v_k)

式v_jの変数v_kの係数をv_iとする．

例．　$y=5a^2+(6a+1)a^2$ のaの係数を求める．

XA=A**2

YA=(5*XA)+(((6*A)+1)**A)

XC=COEF(YA,XA)

結果　XC=6+(6*A)

viii) 単　純　化

SIMPLIFY v_i

式v_iを単純化する．

この場合の単純化は，表1の単純化，同類項のまとめ，正規化のことである．

例.　Y=(2*(X**2))+(((X**2)+5)/1)

SIMPLIFY Y

結果　Y=3*(X**2)+5

式を展開したり，因数分解したりした方が単純だと思われる場合はEXPAND，COEF などの命令を使うと良い．SIMPIFY では，その種の単純化は行わない．

ix) 印　刷

PRINT v_i

式 v_i をタイプライターに印刷する．

PRINT+ v_i

式 v_i のすべての式変数を原子変数だけの式に置き換えて印刷する．

例.　ATOMIC I,J,K

X=(I+J)*(Y/K)

Y=SIN(I*J)

PRINT+ X

結果　X=(I+J)*(SIN(I*J)/K) と印刷される．

PRINT X

結果　X=(I+J)*(Y/K)　と印刷される．

③ 数式の内部表示

数式は内部では二進木リスト構造に変換される．

例.　X＝A＋5の内部表示

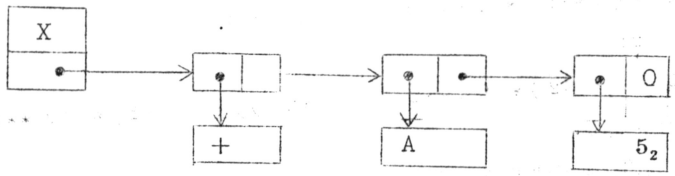


図　2

数式処理中に新しいリストセルが必要になると，未使用リストセルのリストから切り取つて使用する．また，数式処理中に不要となつたリストセルは，そのことを表す旗を立てて切り離される．そのようなリストセルは，未使用リストセルのリストがなくなつたとき，ゴミ集めルーテインによつて再び未使用リストセルとして使用できるようになる．

各リストセルには，そのリストセルの持つている記号の入つている番号地があるので，原子変数，式変数のために，変数をBCDの形で貯えておく表がある．この表は，ある変数が原子変数であるか，式変数であるかを判別したり，ある式変数に対する式のリストが

どこにあるかを知るために使われる.

④ 使 用 例

a. 行列の積 C＝A ＊Bを求める.

$$A=\begin{bmatrix} X-Y & 1 \\ X^2 & X+Y \end{bmatrix} \qquad B=\begin{bmatrix} X+Y & X^2-Y^2 \\ 2 & X^3-Y^3 \end{bmatrix}$$

```
ATOMIC  X,Y
A11=X-Y
A12=X**2
A21=1
A22=X+Y
B11=X+Y
B12=2
B21=(X**2)-(Y**2)
B22=(X**3)-(Y**3)
C11=(A11*B11)+(A12*B21)
C12=(A11*B12)+(A12*B22)
C21=(A21*B11)+(A22*B21)
C22=(A21*B12)+(A22*B22)
PRINT+C11
```

結果 C11=((X-Y)*(X+Y))+2

```
PRINT+ C12
```

結果 C12=((X-Y)*((X**2)-(Y**2)))+((X**3)-(Y**3)).

```
PRINT+ C21
```

結果 C21=((X**2)*(X+Y))+((X+Y)*2).

```
PRINT+ C22
```

結果 C22=((X**2)*((X**2)-(Y**2)))+((X+Y)*((X**3)-
(Y**3))).

ここでのcijは答にあるような原子変数への置換はまだされていないことに注意.

b. 2次方程式の根を求める. ($AX^2+BX+C=0$として, A=1, B=4, C=4)

```
SOLQUAD1=((-B)+(((B**2)-(4*(A*C)))**(0.5)))/(2*4).
SOLQUAD2=((-B)-(((B**2)-(4*(A*C)))**(0.5)))/(2*4).
ANS1=EVAL SOLQUAD1(A,1)(B,4)(C,4)
ANS2=EVAL SOLQUAD2(A,1)(B,4)(C,4)
```

結果　ANS1＝－2

ANS2＝－2

この例で，答が複素数になるような場合は，システムが計算できないため，EVALのステートメントを打つた後，ただちにエラーメッセージが出てくる．そのため，例外処理のプログラムは必要ない．

c．式のまとめ

$Z=ax+at^2x+at^3x^2+abtx^2+astx^3$ を a と x について因数分解する．

```
ATOMIC  A,X,S,T
X2=X**2
X3=X**3
Z=(((A*X)+((A*(T**2))*X))+((A*(T**3))*X2))
  +((((A*B)*T)*X2)+(((A*S)*(T**3))*X3))
ACOEFF=COEF(Z,A)
```

結果　ACOEFF＝X＋T²X＋T³X2＋BTX2＋ST³X3

```
XCOEFF=COEF(ACOEFF,X)
```

結果　XCOEFF＝1＋T²

```
X2COEFF=COEF(ACOEFF,X2)
```

結果　X2COEFF＝T³＋BT

```
X3COEFF=COEF(ACOEFF,X3)
```

結果　X3COEFF＝ST³

```
ANS=A*(((X*XCOEFF)+((X**2)*X2COEFF))
     +((X**3)*X3COEFF)).
```

結果　ANS＝$a(x(1+t^2)+x^2(t^3+Bt)+x^3(st^3))$．

d．微　分

$(ax/b+\cos(x))'$ を求める．

```
ATOMIC  A,X,B
Y=(A*X)/(B+COS(X))
YD=DIF(Y,X)
```

結果　YD＝－(A/(SIN(X)))．

e．積　分

$\int(\sin(kx)-ax)dx$ を求める．

```
YY=(SIN(K*X))-(A*X)
YYI=INTEG(YY,X)
```

結果　YYI＝(－(COS(K*X))/K)－(A*((X**2)/2))．

おわりに，いろいろ御指導いただいた浦教授およびライブラリールーテイン，数値変換ルーテインを提供していただいた中西君に感謝いたします．

本 PDF ファイルは 1968 年発行の「第 9 回プログラミング一シンポジウム報告集」をスキャンし、項目ごとに整理して、情報処理学会電子図書館「情報学広場」に掲載するものです。

この出版物は情報処理学会への著作権譲渡がなされていませんが、情報処理学会公式 Web サイトの https://www.ipsj.or.jp/topics/Past_reports.html に下記「過去のプログラミング・シンポジウム報告集の利用許諾について」を掲載して、権利者の捜索をおこないました。そのうえで同意をいただいたもの、お申し出のなかったものを掲載しています。

過去のプログラミング・シンポジウム報告集の利用許諾について

情報処理学会発行の出版物著作権は平成 12 年から情報処理学会著作権規程に従い、学会に帰属することになっています。

プログラミング・シンポジウムの報告集は、情報処理学会と設立の事情が異なるため、この改訂がシンポジウム内部で徹底しておらず、情報処理学会の他の出版物が情報学広場 (=情報処理学会電子図書館) で公開されているにも拘らず、古い報告集には公開されていないものが少からずありました。

プログラミング・シンポジウムは昭和 59 年に情報処理学会の一部門になりましたが、それ以前の報告集も含め、この度学会の他の出版物と同様の扱いにしたいと考えます。過去のすべての報告集の論文について、著作権者（論文を執筆された故人の相続人）を探し出して利用許諾に関する同意を頂くことは困難ですので、一定期間の権利者捜索の努力をしたうえで、著作権者が見つからない場合も論文を情報学広場に掲載させていただきたいと思います。その後、著作権者が発見され、情報学広場への掲載の継続に同意が得られなかった場合には、当該論文については、掲載を停止致します。

この措置にご意見のある方は、プログラミング・シンポジウムの辻尚史運営委員長 (tsuji@math.s.chiba-u.ac.jp) までお申し出ください。

加えて、著作権者について情報をお持ちの方は事務局まで情報をお寄せくださいますようお願い申し上げます。

期間：2020 年 12 月 18 日 〜 2021 年 3 月 19 日
掲載日：2020 年 12 月 18 日

プログラミング・シンポジウム委員会

情報処理学会著作権規程
https://www.ipsj.or.jp/copyright/ronbun/copyright.html