寄　　書

数値計算を最終目標とした数式処理*

石　黒　美　佐　子**

1. まえがき

最近国内でも，数式処理に属する分野での計算機使用がふえてきた．しかし，その興味はもっぱら数式処理により得た数式そのものにあって，それに数値をあてはめて，いわゆる数値計算を実行することを目的とした試みは少ない．現在する数式処理用言語も，IBM 7090/94 用の FORMAC ように，インタプリティブな方法で数値計算を実行できるものもあるが，重点はやはり数式を導き出すことに置かれている．

ここでの目的は，導き出された数式を，FORTRAN のプログラムに自動的に変換し，数値計算に役立たせることである．その際に，一番問題となるのは，得た数式が長すぎて，1つの FORTRAN ステートメントにおさまらない場合に，いくつかのステートメントに分割する必要が生じることである．以下で，このことに重点を置いて，処理方法と適用例，さらに，すでにある FORMAC との関係について述べたい．

2. 方　　法

2.0 処理言語

処理言語は，FORTRAN を主とし，リスト処理や文字取り扱いには SLIP を使用した．LISP は入出力形式が S-式で，結果を数値計算に結びつけにくい点に問題がある．もし FORMAC が使用可能なら，FORTRAN に近い形で数式が出力されるので，処理過程のうち次節の第1段階は省略できる．

2.1 数式処理

式の導き出し方の詳細は，触れないことにする．数式処理により得た数式の内部表現は，リスト構造をなすが，これを FORTRAN に近い形式で出力することに重点をおく．すなわち，数式の構成要素の空白をつめて，ひとまず大きな array にパックする．

2.2 1つの FORTRAN ステートメントにおさまらない数式の処理

準備として，前節の数式処理過程と並行に実行できるが，数式を作成しながら，数式の構成文字に初めから番号をつけ，この文字番号に基づいて，数式の括弧の位置，切断可能な時点等式の構造をチェックし保存する．数式を切断するやり方は，括弧の中をさきに計算するとか，＋－記号が出てきた時点でステートメントを分けるとか，われわれが通常行なっている方法を採用する．文字番号は，次の4つの時点で保存される．

（1）ステートメントの終わり．

（2）＝　　：数式の右辺と左辺の区別．

（3）＋，－：数式はこの時点で切断可能．

（4）（，）　：括弧の位置(サブスクリプトは除く)．

括弧内を1つの数式とみなし，さきに計算することが可能であるが，この場合に，これがまた長くて，1つの FORTRAN ステートメントにおさまらない場合に，分割することもありうる．したがって，括弧の深さにより，式は木構造をなし，リカーシブな取扱いが必要である．

本過程では，前段階で作成された数式を，その構造に基づいて，切断し．さらに1つの数式の構成部分としての関連を保たせるために，バラバラになった式を接続するための“つなぎ”を入れる．これを自動的に行なうには，次の3つの条件を考慮しなければならない．数式の左辺を x で表わし，分割された式の断片を P で表わそう．

1. 数式の括弧のネストの深さ (l)

　$l=0$ のとき　$x=\cdots\cdots$,

　$l=1$　　$\mathrm{temp}_1=\cdots\cdots$,

　$l=2$　　$\mathrm{temp}_2=\cdots\cdots$.

　⋮

* Symbolic Maritulation for Nnmerical Computation, by Misako Ishiguro (Japan Atomic Energy Research Institute, Computing Center)

** 日本原子力研究所

2. P が数式の最初の部分か（f=0 or 1）

   f=0 のとき　$x=P$，

   $temp_l=P$，

   f=1　例　$x=x+P$，

   $temp_l=temp_l+P$

3. 括弧内の計算（P'）が先行するか（S=0 or 1）

   S=1 のとき　$temp_{l+1}=P'$，

   例　$temp_l=temp_{l+1}\div P$．

上記の条件により，ステートメントの作り方は次の6通りである．

1. $l=f=S=0$ のとき

   $x=P$

2. $l=0$, $f=1$, $S=0$

   例　$x=x+P$

3. $l=0$, $f=S=0$

   例　$x=x+temp_1*P$

4. $l>1$, $f=S=0$

   $temp_l=P$

5. $l>1$, $f=1$, $S=0$

   例　$temp_l=temp_l+P$

6. $l>1$, $f=S=1$

   例　$temp_l=temp_l-temp_{l+1}*P$

このようにして数式をFORTRANステートメントの制限内におさまるようにいくつかに分割し，それらを有機的に結合し，もとの一つの数式を構成する．

2.3 FORTRAN ステートメントの自動作成

この過程で，分割された式を，次の手順でFORTRANステートメントのカードイメージに変換する．この作業を前節の過程と並行して行なう．

1. カードの第6欄に，continuation のカードシシーケンス番号を入れる．
2. 数式を，第7欄から第72欄までにつめる．
3. 1つのステートメントが第72欄までにおさまらないときは，1, 2をくりかえす．
4. 1, 2をくりかえすうちに，主記憶にあらかじめ用意したステートメントの保存場所（カード10枚分程度）がいっぱいになったら補助記憶に掃き出す．
5. 1つのステートメントが終われば，カードの途中なら残りに空白を入れる．
6. 1つのプログラムが終われば，主記憶に残っているステートメントを補助記憶に掃き出す．

```
LL=  1700
LST
PHI(5)=I(5)*EXP(P(5)*T)+A(5,4)*I(4)*(G(4,5)*EXP(P(5)*T)+G(5,
4)*EXP(P(4)*T))+A(5,3)*I(3)*(G(3,5)*EXP(P(5)*T)+G(5,3)*EXP(P
(3)*T))+A(5,2)*I(2)*(G(2,5)*EXP(P(5)*T)+G(5,2)*EXP(P(2)*T))+
A(5,1)*I(1)*(G(1,5)*EXP(P(5)*T)+G(5,1)*EXP(P(1)*T))+A(5,4)*A
(4,3)*I(3)*(G(4,5)*G(3,5)*EXP(P(5)*T)+G(5,4)*G(3,4)*EXP(P(4)
*T)+G(5,3)*G(4,3)*EXP(P(3)*T))+A(5,4)*A(4,2)*I(2)*(G(4,5)*G(
2,5)*EXP(P(5)*T)+G(5,4)*G(2,4)*EXP(P(4)*T)+G(5,2)*G(4,2)*EXP
(P(2)*T))+A(5,4)*A(4,1)*I(1)*(G(4,5)*G(1,5)*EXP(P(5)*T)+G(5,
4)*G(1,4)*EXP(P(4)*T)+G(5,1)*G(4,1)*EXP(P(1)*T))+A(5,3)*A(3,
2)*I(2)*(G(3,5)*G(2,5)*EXP(P(5)*T)+G(5,3)*G(2,3)*EXP(P(3)*T)
+G(5,2)*G(3,2)*EXP(P(2)*T))+A(5,3)*A(3,1)*I(1)*(G(3,5)*G(1,5
)*EXP(P(5)*T)+G(5,3)*G(1,3)*EXP(P(3)*T)+G(5,1)*G(3,1)*EXP(P(
1)*T))+A(5,2)*A(2,1)*I(1)*(G(2,5)*G(1,5)*EXP(P(5)*T)+G(5,2)*
G(1,2)*EXP(P(2)*T)+G(5,1)*G(2,1)*EXP(P(1)*T))+A(5,4)*A(4,3)*
A(3,2)*I(2)*(G(4,5)*G(3,5)*G(2,5)*EXP(P(5)*T)+G(5,4)*G(3,4)*
G(2,4)*EXP(P(4)*T)+G(5,3)*G(4,3)*G(2,3)*EXP(P(3)*T)+G(5,2)*G
(4,2)*G(3,2)*EXP(P(2)*T))+A(5,4)*A(4,3)*A(3,1)*I(1)*(G(4,5)*
G(3,5)*G(1,5)*EXP(P(5)*T)+G(5,4)*G(3,4)*G(1,4)*EXP(P(4)*T)+G
(5,3)*G(4,3)*G(1,3)*EXP(P(3)*T)+G(5,1)*G(4,1)*G(3,1)*EXP(P(1
)*T))+A(5,4)*A(4,2)*A(2,1)*I(1)*(G(4,5)*G(2,5)*G(1,5)*EXP(P(
5)*T)+G(5,4)*G(2,4)*G(1,4)*EXP(P(4)*T)+G(5,2)*G(4,2)*G(1,2)*
EXP(P(2)*T)+G(5,1)*G(4,1)*G(2,1)*EXP(P(1)*T))+A(5,3)*A(3,2)*
A(2,1)*I(1)*(G(3,5)*G(2,5)*G(1,5)*EXP(P(5)*T)+G(5,3)*G(2,3)*
G(1,3)*EXP(P(3)*T)+G(5,2)*G(3,2)*G(1,2)*EXP(P(2)*T)+G(5,1)*G
(3,1)*G(2,1)*EXP(P(1)*T))+A(5,4)*A(4,3)*A(3,2)*A(2,1)*I(1)*(
G(4,5)*G(3,5)*G(2,5)*G(1,5)*EXP(P(5)*T)+G(5,4)*G(3,4)*G(2,4)
*G(1,4)*EXP(P(4)*T)+G(5,3)*G(4,3)*G(2,3)*G(1,3)*EXP(P(3)*T)+
G(5,2)*G(4,2)*G(3,2)*G(1,2)*EXP(P(2)*T)+G(5,1)*G(4,1)*G(3,1)
*G(2,1)*EXP(P(1)*T))
```

第 1 図

3. IBM 7044 での適用例

次の軟化式

$$\varphi_n(k,p)=G_n(k,p)\left\{I_n(k)+\sum_{l=1}^{n-1}a_{n,l}\varphi_l(k,p)\right\},$$

$$\varphi_1(k,p)=G_1(k,p)I_1(k),$$

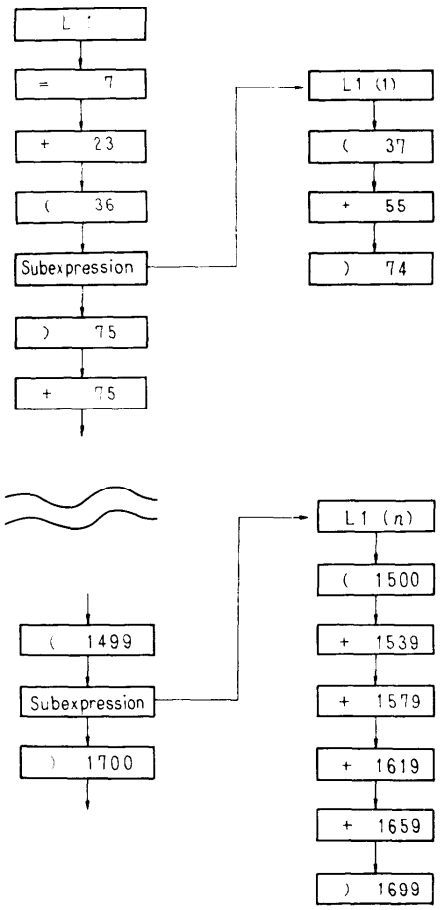
$$G_n(k,p)=\frac{1}{p+c_n(b_n+k^2)}$$

の逆ラプラス変換

$$\phi_n(k,t)=L^{-1}\varphi_n(k,p).$$

を数式的に求め，得られた結果に数値をあてはめて，くりかえし計算したい場合を考えてみる．

数式処理した式は，第1の過程で $n=5$ の場合を例にとると**第1図**のように出力する．これと並行に，数式の構成に関する情報を**第2図**のように木構造で保存



第 2 図

する．$n=5$ の場合は，括弧のネストの深さは2である．この数式は，1,700 字でできていてもちろん1つの FORTRAN ステートメントにはおさまらない．したがって，切断して，**第3図**で示すように5つのステートメントに分割する．実際の FORTRAN デックは，これらのステートメントの他に，コントロールカード，サブルーチンの定義，DIMENSION を最初に，RETURN, END を後に，必要なものを自動的につけ加える．これを，オペレーティングシステムで規定された形式で補助記憶にいったん保存し，カード等で入力した他のプログラムデックと結合し，数値計算を可能にする．

4. FORMAC との関係

FORMAC には，EVAL 以下の演算を実行するステートメントは備わっている．その上 ATOMIC と declare されない変数はみんな数値として扱われるので，数式に数値を入れて計算する機能はある．しかしその処理の方法は，ポーリシュ・フォームに内部表現した式をインタプリティブに計算するので，得られた数式をくりかえし数値計算に使用するのは，演算の効率が悪い．数値計算を最終目的とする場合は，FORTRAN 言語等でカードイメージにいったん出力し，以後数値計算に使用できれば好都合である．たとえば，FORMAC に DECK, COMPUTE 機能を付加し，いままでの FORMAC の仕様で，数式 $F(x_1,x_2,\cdots,x_n)$ を作成したら，DECK $F(x_1,x_2,\cdots,x_n)$ 命令により，関数 $F(x_1,x_2,\cdots,x_n)$ が，独立のサブルーチンとして作成され，以後それにいろいろなデータをあてはめて計算したいときは，LET A=COMPUTE $F(v_1,v_2,\cdots,v_n)$ とすれば，普通のサブルーチンと同じような呼び出し方で，計算の効率を落さずに，数値計算が実行できるとよい．

5. ま と め

以上で，数値計算の効率を落さないことを主眼として，数式処理を自動的に数値計算に結びつけることに関して，一応その目的は達した．反省すべき点として，DO LOOP 等をうまく使ってプログラムを作成できれば，プログラムの長さがもっと短くなるが，この適応例では，数式を次々と書き並べただけなので，$n=8$ の場合でさえ，1回でコンパイルできない長いプログラムデックになってしまうことである．

数式処理の結果が実用に寄与できるという大きな目

```
 IF(NMAX.LE.5)RETURN
 TMP1 =G(3,5)*G(1,5)*EXP(P(5)*T)+G(5,3)*G(1,3)*EXP(P(3)*T)+G(5,1)*G
1(3,1)*EXP(P(1)*T)
 PHI(5)=I(5)*EXP(P(5)*T)+A(5,4)*I(4)*(G(4,5)*EXP(P(5)*T)+G(5,4)*EXP
1(P(4)*T))+A(5,3)*I(3)*(G(3,5)*EXP(P(5)*T)+G(5,3)*EXP(P(3)*T))+A(5,
22)*I(2)*(G(2,5)*EXP(P(5)*T)+G(5,2)*EXP(P(2)*T))+A(5,1)*I(1)*(G(1,5
3)*EXP(P(5)*T)+G(5,1)*EXP(P(1)*T))+A(5,4)*A(4,3)*I(3)*(G(4,5)*G(3,5
4)*EXP(P(5)*T)+G(5,4)*G(3,4)*EXP(P(4)*T)+G(5,3)*G(4,3)*EXP(P(3)*T))
5+A(5,4)*A(4,2)*I(2)*(G(4,5)*G(2,5)*EXP(P(5)*T)+G(5,4)*G(2,4)*EXP(P
6(4)*T)+G(5,2)*G(4,2)*EXP(P(2)*T))+A(5,4)*A(4,1)*I(1)*(G(4,5)*G(1,5      3
7)*EXP(P(5)*T)+G(5,4)*G(1,4)*EXP(P(4)*T)+G(5,1)*G(4,1)*EXP(P(1)*T))
8+A(5,3)*A(3,2)*I(2)*(G(3,5)*G(2,5)*EXP(P(5)*T)+G(5,3)*G(2,3)*EXP(P
9(3)*T)+G(5,2)*G(3,2)*EXP(P(2)*T))+A(5,3)*A(3,1)*I(1)* TMP1
 TMP1 =G(3,5 )*G(2,5)*G(1,5)*EXP(P(5)*T)+G(5,3)*G(2,3)*G(1,3)*EXP(P
1(3)*T)+G(5,2)*G(3,2)*G(1,2)*EXP(P(2)*T)+G(5,1)*G(2,1)*EXP(P
2(1)*T)
 PHI(5)=     PHI(5)+A(5,2)*A(2,1)*I(1)*(G(2,5)*G(1,5)*EXP(P(5)*T)+G
1(5,2)*G(1,2)*EXP(P(2)*T)+G(5,1)*G(2,1)*EXP(P(1)*T))+A(5,4)*A(4,3)*
2A(3,2)*I(2)*(G(4,5)*G(3,5)*G(2,5)*EXP(P(5)*T)+G(5,4)*G(3,4)*G(2,4)
3*EXP(P(4)*T)+G(5,3)*G(4,3)*G(2,3)*EXP(P(3)*T)+G(5,2)*G(4,2)*G(3,2)      4
4*EXP(P(2)*T))+A(5,4)*A(4,3)*A(3,1)*I(1)*(G(4,5)*G(3,5)*G(1,5)*EXP(
5P(5)*T)+G(5,4)*G(3,4)*G(1,4)*EXP(P(4)*T)+G(5,3)*G(4,3)*G(1,3)*EXP(
6P(3)*T)+G(5,1)*G(4,1)*G(3,1)*EXP(P(1)*T))+A(5,4)*A(4,2)*A(2,1)*I(1
7)*(G(4,5)*G(2,5)*G(1,5)*EXP(P(5)*T)+G(5,4)*G(2,4)*G(1,4)*EXP(P(4)*
8T)+G(5,2)*G(4,2)*G(1,2)*EXP(P(2)*T)+G(5,1)*G(4,1)*G(2,1)*EXP(P(1)*
9T))+A(5,3)*A(3,2)*A(2,1)*I(1)*TMP1
 PHI(5)=     PHI(5)+A(5, 4)*A(4,3)*A(3,2)*A(2,1)*I(1)*(G(4,5)*G(3,5
1)*G(2,5)*G(1,5)*EXP(P(5)*T)+G(5,4)*G(3,4)*G(2,4)*G(1,4)*EXP(P(4)*T
2)+G(5,3)*G(4,3)*G(2,3)*G(1,3)*EXP(P(3)*T)+G(5,2)*G(4,2)*G(3,2)*G(1
3,2)*EXP(P(2)*T)+G(5,1)*G(4,1)*G(3,1)*G(2,1)*EXP(P(1)*T))          5
```

第 3 図

的のために，出力形式をいろいろ考察するのは，有意義なことである．

筆者に例題を提供され，数式のチェックをして下さった原研の小山勤二氏に感謝する．

参考文献

1) 石黒美佐子：逆ラプラス変換の数式処理による試み，JAERI-memo 3047.

2) IBM: User's PRELIMINARY REFERENCE MANUAL for the experimental FORmula MAnipulation Compiler (FORMAC).

3) Weizenbaun: Symmetric List Processor, Comm. ACM (6-9), p. 524～544.

(昭和 43 年 5 月 4 日受付，昭和 43 年 9 月 13 日再受付)