C 2. 数式処理における二，三の試み

黒沢俊雄，前田英次郎，藤井　宏，志水敬子（三菱原子力）

§0 序

数値計算技術の進歩と数値計算用プログラム言語の開発によつて，数値計算を行うために計算機を用いることが可能になつた．しかし，記号の数学はどうであろうか．過去数年にわたつて，多くの努力がこの方面に払われてきた．その一つがFORMACの開発であつた．FORMACは数式処理用のプログラム言語でいくつかの数式操作命令をふくんだシステムである．又，Lispのような汎用記号処理システムなども開発されてきた．

この論文では，これらのシステムの詳細をのべることはさけ，これらのシステムとEORTRANを応用したいくつかの数式処理の実例をあげ，数式処理の可能性と問題点をさぐつていただくための資料を提供することにした．

§1 Matrix Inversion

元がsymbolic expressionである行列の逆行列を求めるprogramをFORMACを用いて作つた．行列が4×4程度の問題であればin-coreで解くこともできよう．しかし，このprogramは10×10程度のものをねらい，容量を考えてつぎのようにprogramした．すなわち，

$$A^{-1}=\frac{1}{|A|}\begin{pmatrix} A_{11} & \cdots\cdots & A_{n1} \\ & & \\ A_{1n} & \cdots\cdots & A_{nn} \end{pmatrix} \qquad A_{ij} \text{ は } a_{ij} \text{ の余因数}$$

において，A^{-1} の(i,j)を求めてはout putする方法を取つた．
$|A|$ を求めるroutineは行列式の展開の方法をprogram化したものである．

§2 Striffness Matrix

$$K=B^{T}DB \quad \cdots\cdots \quad (1)$$

Kを求める問題において，Bが(2)の型をしたmatrixであるから，FORMACを使用して，matrix Bを計算し，その後FORTRANにうけついだ．

$$B=\begin{bmatrix} -2^{2}F_{i}/2x^{2} \\ -2^{2}Fi/2y^{2} \\ -2^{2}Fi/2x2y \end{bmatrix} \quad \cdots\cdots \quad (2)$$

$$i=x_1\,y_1\,x_2\,y_2\,x_3\,y_3$$

F_{xi} F_{yi} の型は下のような複雑な形をしている.

$$F_{x_1}=(Y_2-Y_1)\left(L_1^2L_2+\frac{1}{2}L_1L_2L_3\right)+(Y_3-Y_1)\left(L_1^3L_2+\frac{1}{2}L_1L_2L_3\right)$$

$$F_{x_2}=(Y_3-Y_2)\left(L_2^2L_3+\frac{1}{2}L_1L_2L_3\right)+(Y_1-Y_2)\left(L_2^2L_1+\frac{1}{2}L_1L_2L_3\right)$$

$$\vdots$$

$$F_{y_3}=(X_3-X_1)\left(L_3^2L_1+\frac{1}{2}L_1L_2L_3\right)+(X_3-X_2)\left(L_3^2L_1+\frac{1}{2}L_1L_2L_3\right)$$

$$L_1=(A_1+B_1x+C_1y)/2\Delta$$
$$L_2=(A_2+B_2x+C_2y)/2\Delta \qquad 2\Delta=\begin{vmatrix}1 & X_1 & Y_1\\ 1 & X_2 & Y_2\\ 1 & X_3 & Y_3\end{vmatrix}$$
$$L_3=(A_3+B_3x+C_3y)/2\Delta$$

上式であきらかなようにBはx^1, y^1, constantの三項からのみなつている.
よつてFORMACでそれぞれの係数を取り出し, FORTRANのDataとした.

§ 3.0 原子核の三体問題

proton-neutron-neutron, proton-proton-neutron
の三体問題のRayleigh-Ritz法による波動関数

Rayleigh-Ritz法によつて, 原子核の三体問題の, 固有波動函数を求めることが問題である.

次の2次形式の係数を計算する.

$$\int \Psi \cdot A\Psi \, d\tau = \sum_{i,j} a_{ij} x_i x_j$$

$\Psi(r_1, r_2, r_3)$は, 次の大村型波動函数の形を与える.

$$\Psi = \sum_i x_i \cdot \Psi_i$$

$$\Psi_i = \prod_{j=1}^{3}\left(e^{-\mu(r_j-D)} - e^{-\mu(r_j-D)}\right) \cdot P_i(r_1, r_2, r_3)$$

$$P_1 = 1$$
$$P_2 = r_1+r_2$$
$$P_3 = r_1^2+r_2^2$$
$$P_4 = (r_1+r_2)r_3$$
$$P_5 = r_3$$
$$P_6 = r_3^2$$
$$P_7 = r_1r_2$$

$$P_8 = r_2-r_1$$
$$P_9 = r_2^2-r_1^2$$
$$P_{10} = (r_2-r_1)r_3$$
$$P_{11} = r_1^2r_2-r_1r_2^2 + r_2^2r_3-r_2r_3^2 + r_3^2r_1-r_3r_1^2$$

演算子Aは次のものがある．

$$N = 1$$

$$K = \sum_{(i,j,k)J} \frac{1}{r_i^2} \cdot \frac{\sigma}{\sigma r_i}\left(r_i^2 \frac{\sigma}{\sigma r_i}\right) + \frac{r_j^2 + r_k^2 - r_i^2}{2 r_j r_k} \cdot \frac{\sigma^2}{\sigma r_j \sigma r_k}$$

$$J = \{(1,2,3),(2,3,1),(3,1,2)\}$$

$$C = \frac{1.44}{r^2}\left[1 - \left(1 + \frac{11}{16}\beta r_3 + \frac{3}{16}\beta^2 r_3^2 + \frac{1}{48}\beta^3 r_3^3\right) \cdot e^{-\beta r_3}\right]$$

$$U_i^M = \int \Psi_s \left\{u_i(r_3) p_{12} + \frac{1}{4} u_i(r_1) p_{23} + \frac{1}{4} u_i(r_2) p_{12}\right\} \Psi_s d\tau$$

p_{ij} ： 変数r_i とr_j の交換operator

$$u_i(r) = \exp\{-\delta_i (r - D)\},$$

他に

$$U_i^W \ , \ P \ , \ E_i \ ,$$

等の演算子の計算が必要である．

積分 $\int f \cdot d\tau$ は次式で定義される．

$$\int f(r_1, r_2, r_3) d\tau = \int_D^\infty r_1 dr_1 \int_D^\infty r_2 dr_2 \int_D^\infty r_3 \cdot f(r_1 r_2 r_3) dT_3$$
$$- \int_D^\infty r_2 dr_2 \int_D^\infty r_3 dr_3 \int_0^\infty r_1 \cdot f(r_1 r_2 r_3) du_1$$
$$- \int_D^\infty r_3 dr_3 \int_D^\infty r_1 dr_1 \int_0^\infty r_2 f(r_1 r_2 r_3) du_2$$
$$- \int_D^\infty r_1 dr_1 \int_D^\infty r_2 dr_2 \int_0^\infty r_3 \cdot f(r_1 r_2 r_3) du_3$$

$$u_1 = r_1 - r_2 - r_3$$
$$u_2 = r_2 - r_3 - r_1$$
$$u_3 = r_3 - r_1 - r_2$$

数式処理の問題としては，微分演算，$K\Psi$を行う事と，積分$\int f d\tau$ を行うことが必要である．

微分演算

$$K \cdot \Psi$$

に対しては，LISP，FORMAC，FORTRAN でそれぞれ行つた．

積分 $\int f d\tau$ は分解すればすべて

$$\int_{\beta}^{\infty} r^{n} \cdot e^{-\alpha r} dr \qquad (\alpha > 0)$$

の形に帰着される．従つて，原理的には，数式処理によつて計算可能であるが，計算量は厖大である．

$\int \Psi \cdot N \cdot \Psi \, d\tau$ 等の簡単な例につき，FORMACによつて計算が行われた．

全体として，LISP，FORMAC でいろいろ試みられたが，最終的には，FORTRAN によつて数式処理プログラムを作つて，問題が解かれた．

§ 3.1 LISPによる試み

1) LISP では(1)式を計算させた．

$$\Phi = 4K4 \quad \cdots\cdots\cdots\cdots \quad (1)$$

ここで，Kは微分operator

はじめに作つたprogramはmanual * を参照したもので，入出力の式の表現に下記のようにいくつかの問題点があつた．

イ．1つのoperatorをはさんで2つのargumentがあること．そして，これらの3つの記号をカツコで包まなければならないこと．

ロ．0の消去や同類項のまとめなど，式のsimplificationが行なわれないこと．

ハ．＋，－記号が原子記号として使用出来ず，＝＋，＝－とおきかえなければならないことと．

以上のように入出力する式がやたらにカツコが多くなつたり，項の数が多くなる．

一応programとしては，

① diff ； 微分program

② edit ； simplification

③ 代 入 ； 同類項を1つの変数におきかえるprogram

④ その他

をつくつたが，取り扱いが繁雑で，このままでは積分をすることが困難であつた．したがつて，微分積分を行いやすいような式の表現を工夫しなければならず，つぎのような表現を考えた．

---

* 情報処理言語LISP：電子協編
Advance in programming and non-numeric computation:
Perganon Press.

1. 多項式のリスト表現

$$A=\sum_i X_i \qquad X_i=\prod_j a_i x_j^{e_j} \quad \cdots\cdots\cdots (2)$$

なる多項式をつぎのようなリスト表現におきかえした.

$$A=(X_1 X_2 \cdots\cdots X_n)$$

$$X_i=[(x_1 e_1)(x_2 e_2)\cdots\cdots a_i)$$

2. このリスト表現の中で変数間のorderにpriorityをあたえsortした.
さらに巾に関しては昇巾の順とした.
(3)式におけるorderのpriorityは,

$$(x_1 x_2 \cdots\cdots x_n)$$

である.

<例> $2x^3+3y^3+z^3+7x^2y-xz^2+xy^2+xyz+3 \cdots\cdots (4)$

```
( ((X3)2)((X2)(Y1)7)
  ((X1)(Y2)1)((X1)(Y1)(Z1)1)
  ((X1)(Z2)-1)((Y3)3)
  ((Z3)1) (3)  )
```

priority list は(XYZ)

2) 三体問題にはつぎのroutineが必要である.

① add 加算
② mult 乗算
③ subtract 減算
④ diff 微分
⑤ adjust 数式をorderの順に並び変える.
⑥ exchange 数式中の二変数を交換する.
⑦ subst 数式中の一変数に多項式を代入する.
⑧ 積 分
⑨ output

現在1～7のroutineを完成している, ⑧⑨に関しては上の7つのroutineをtest中時間及びcoreの問題が生じた為, 未完成である.

3) 計算所用時間とスペースに関して
どの程度の問題で, どの程度の時間と記憶容量が必要であろうか.
そこで多項式の展開を行い, 新たな多項式を発生させる実験をしてみた.

その問題で9乗の計算を行つている途中でcoreが一杯になつた．

$$(X+Y+Z+U)^{i}=(X+Y+Z+U)^{i-1}(X+Y+Z+U)$$

i	時間(sec)	項　数
2	1.75	4×4
3	3.03	20×4
4	8.45	35×4
5	14.55	56×4
6	26.55	84×4
7	43.17	120×4
8	68.4	165×4

§ 3.2　FORMACによる試み

LISPでの試みが時間と記憶容量の点で不可能らしいとわかつたころ，FORMACのことを知り間もなく我々の所でも使用できるようになつた．

FORMACには，微分の機能があるので，A4の計算は問題なくできる．積分の機能は含まれていないが，数式を分解して，特定の項や特定の因子を取出すことができる．従つて，LISPでもやつたように，被積分函数を分解して

$$\int_b^\infty r^n e^{-ar}dr=\frac{b^n}{a}e^{-ab}+\frac{n}{a}\int_b^\infty r^{n-1}e^{-ar}dr$$

$$\int_b^\infty e^{-ar}dr=\frac{1}{a}e^{-ab}$$

を適用することができる．

LISPと比較して，実用上最大の利点は，FORTRAN Ⅳ の拡張として設計されているので，記憶容量が不足すれば，外部記憶装置を利用できることである．数式の内部表現をそのまま外部記憶装置に出すことはできないが，数式を文字表現に直したものならば，FORTRANの範囲でI/Oができる．

FORMACで試みた結果は最も簡単な場合，

$$\int\phi_1\phi_1\,d\tau=\int\left\{\prod_{k=1}^{3}\left(e^{-\mu(r_k-D)}-e^{-\nu(r_k-D)}\right)\right\}^2 d\tau$$

$$=\int\sum_e C_e\exp\left\{\sum_{k=1}^{3}\alpha_{ke}(r_k-D)\right\}d\tau$$

$$=\sum_e C_e\int\exp\left\{\sum_{k=1}^{3}\alpha_{ke}(r_k-D)\right\}d\tau$$

の1つの l の計算に12秒程度かかり，$l=1\sim36$ なので，上式を計算するのに約7分かかることになる．この程度のスピードならば人海戦術の方が幾分安くつくだろう．

§ 3.3 FORTRAN による試み

LISP, FORMAC での試みが何れも時間の点で実用にならないので，数式処理的なルーチンをFORTRAN で書いて計算することになつた．

積分をパラメーターを含んだ一般式として求めることはあきらめて，先ず積分変数 $r_1\ r_2 r_3$ 以外のパラメータには全てあらかじめ数値を与えることにする．従つて，積分の結果は一つの数値になる．パラメータの値を変えることによつて，いくつかの場合を計算する．

1) 被積分函数は，

$$P(r_1\ r_2\ r_3)\ \exp\left\{\sum_{k=1}^{3}\alpha_k(r_k-D)\right\}$$

($P(r_1\ r_2\ r_3)$ は $r_1\ r_2\ r_3$ の多項式)

の形の項の和と考えて，これを単位として扱う．

2) 多項式は計算機内で次のように表現する．

例．

$$P=r_1\ r_2^{\,2}-3r_2^{\,3}r_3+5r_1-2$$

```
REAL COEF
INTEGER LENGTH, POWER(10,3)
```

LENGTH

4

COEF

+1	-3	5	-2

POWER

1	0	1	0
2	3	0	0
0	1	0	0

3) 演算子が $\boldsymbol{N}$, $\boldsymbol{K}$ の場合には，

$$\int_{\beta}^{\infty} r^n e^{-\alpha r}\,dr$$

の中の (α , β) の組合せが8種類に限られるので，$n=0\sim9$ の場合を全組合せについて計算して表にしておく．

4) $\boldsymbol{K}$ の場合は，微分を必要とするが，

$$\boldsymbol{K}\left\{P_i(r_1, r_2, r_3)\exp\left\{\sum_{k=1}^{3}\alpha_k(r_k-D)\right\}\right\}$$

$$= Q_i(r_1, r_2, r_3, \alpha_1, \alpha_2, \alpha_3) \times \exp\left\{\sum_{k=1}^{3} \alpha_k (r_k - D)\right\}$$

$$= \sum_e R_{ie}(\alpha_1, \alpha_2, \alpha_3)\, r_1^{n_1 e} r_2^{n_2 e} r_3^{n_3 e} \times \exp\left\{\sum_{k=1}^{3} d_k (r_k - D)\right\}$$

(R_{ie} は $\alpha_1 \alpha_2 \alpha_3$ の多項式)

と書けるから，$i=1\sim10$ について，前もって R_{ie} を計算するルーチンを作り，FORTRAN statement card にパンチして使用した．

§4 結　び

数式の計算を計算機でやらせようとすると色々問題点がある．それは上の2，3の例からも分かる通りである．計算機に大学受験の数学の能力をつけようとするには，計算機そのものの機能及び構造を改革しなければならないでしょう．

まず，記憶の形式がもんだいであり，現在の計算機はアレーを中心としたものであり，マトリックスのとりあつかいまでは，自在に操作出来る．たてよこの単純な構造であるからである．しかるに数式は，自然言語ほどではないが，複雑な文法をもつた表現であるので，その処理のむずかしさがある．したがつて，とりあつかう式の意味を整理して，前のこころみでのべた多項式と指数関数のくみあわせのような工夫をすれば，現在の計算機でも充分いけるし，又，FORTRAN などの言語でもプログラムできる．まともに整式表現をそのままの形で処理することは，まだまだ先の話なのではないでしょうか．

いずれの例も 7090 を使用したもので，時間と記憶容量に関する限界については，4項式の展開の実験で，ある程度の予測ができよう．

尚，使用した7090の記憶容量は32K語であるが，LISP などでは約20K語のfree storage が使用可能である．

これから，この種の問題に着手されんとしておられる方に資すれば幸いである．

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