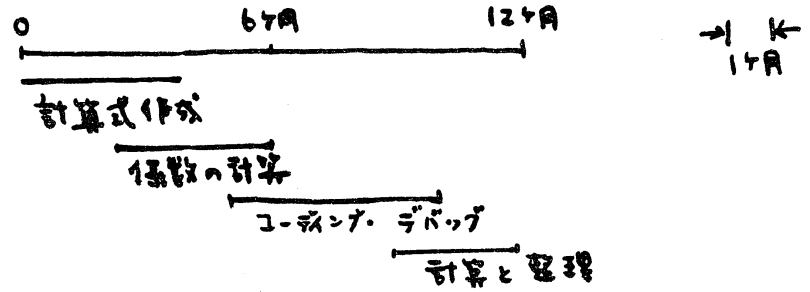
数理解析研究所講究録
109巻 1971年 25-33

数式解析の二.三の試み

航空宇宙技術研究所　戸川隼人

1. なぜ数式解析を始めたか.

「事務と経営」という雑誌に「月産一論文の時代」という話を書いたことがある。(注1) うまく研究管理をすれば平均して月に1編の論文が書ける. ただし大型コンピュータが, かなり自由に使えれば, という話である。まっと旧式の小さなコンピューターを使っていた1960年ごろは, 研究のペースは「年産一論文」に近かった。というのは



というぐらいのペースで, 実際には何人かで分業することにより年産て～3論文を製造していた。

注1) べつに目新しい内容は書いてない。1968年10月号。原題は「ビッグ・サイエンスとコンピュータ」。これには異論もあり, こくのある論文を書こうとすれば「1年に1編以上も論文を出すんはインチキですよ」(東大. 山本善之教授) という意見も真実である。

その時の経験では、過本氏も指摘されたとおり、計算機にかける以前の、式の計算のミスが非常に多く、(もちろん多大の労力もかかり) 長時間かけた計算がムダになったり、原因をさがすのに苦労したりして、なんとかして、ここを機械化して、「確実な処理」ができるようにしたい、ということがあった。

工学上の問題では、基本定理 (基本公式) のようなものがあって、それを個々の小さな具体的な問題に適用していく時に、いつもこの種の問題が起こる。適用分野を限定すれば、それはかなり定形的なパターンであるが、そういうものが実際にはいくつもあるから、どの程度まで一般的な数式処理を行なうべきかは、かなりの自由度がある。

微分方程式の級数展開

パタベーション的な展開

2～4階の微分オペレータの処理 (乗行)

添字のついた複雑な式の処理

$\sum_{\kappa} \phi_{\kappa}(x)$ の積分 (ϕ_{κ} は初等関数) on P

代入、せいとん

このようなものの、いくつかができれば、個々の問題に適用することは可能。他をいえば「単純化に式」のようなものの処理 (自分で定義して) ができるとよい。

2. 2変数多項式の処理

俗に「FORTRANによる数式処理」と呼ばれているものには2種類あって、

- FORTRANで記号処理をやる。
- 多項式の係数のようちょのだけを数値的に扱う

が、やや混同されている。前者については次章3を参照されたい。ここではまず簡単な後者の方から説明する。

[扱った問題]

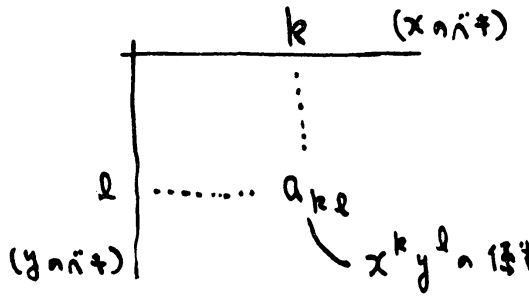
$$J_{ij} \equiv \iint_P \phi_i(x,y)\,\phi_j(x,y)\,dx\,dy$$

を求めること。ただし、ϕ_i, ϕ_j は多項式。

[方針]

項別積分（不定積分）して P の条件を代入する。

[多項式の表現]



$$\phi_i = \sum_k \sum_l a_{kl}\, x^k y^l$$

[カプル－チン]

多項式の積、不定積分、代入

どれも簡単に作れる。しかし能率のいいものはむつかしい。

[問題点]

メモリーを食うこと。コア32Kぐらいでは実用的な問題は解けない(人手でやれる程度まで)。65K語です、たいしたことはできない。

時間がかかること。人間は、Σの一般項のまま計算するので速い(そのかわり、よくまちがえる)。それを、わざわざ、数値的にやるので、なんともムダが多く、HITAC 5020Fです10分とか20分とか(ちょっとした問題でも)それぐらいかかる。ただしプログラムの作り方による。

精度が悪いこと。数百項の加算を何回もくりかえすし、2項係数のようなもので長い値が出ること。大きい項と小さい項が混合していて、一般項で計算すれば大きい項が消えるところを、数値的にやると桁落ちの原因になること、などがあって、結果は、あまり良くなかった。

それでも式の形で積分したから、まだ良かったわけで、数値積分などで処理したら大変なことになる。

[感想]

パラメータを含む2変数多項式を大量に扱うのは無理で、いろいろ問題がある。しかしプログラムは非常に簡単なのでごく小さな問題か、1変数に限って適用すれば有望。三角多項式に応用したという論文も見たことがある。

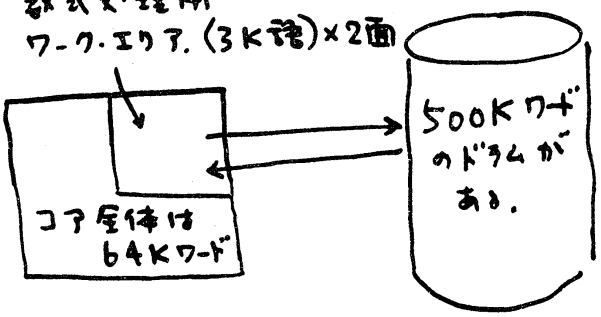
3. 本格的な(?)数式処理の実験

記号処理による、一般的な数式処理のプログラムを作ってみた。まだ虫が多いので、デモンストレーション用オンリーであるが、概要はつぎのとおり。

[機能] 代入、展開、せいとん、単純化、微分、
(もちろん、「ある程度まで」の話)

[対象] 多項式、三角関数、指数(対数)関数、をmixしたもの。微分に関しては1変数のみ。

[使い方] FORTRANのCALL文で呼出す。主な入口は(ユーザが使う)



- ☆ 数式の読込み(オペからコアへ)
- ☆ ドラムに格納(コア → ドラム)
- ☆ ドラムから取出す(ドラム → コア)

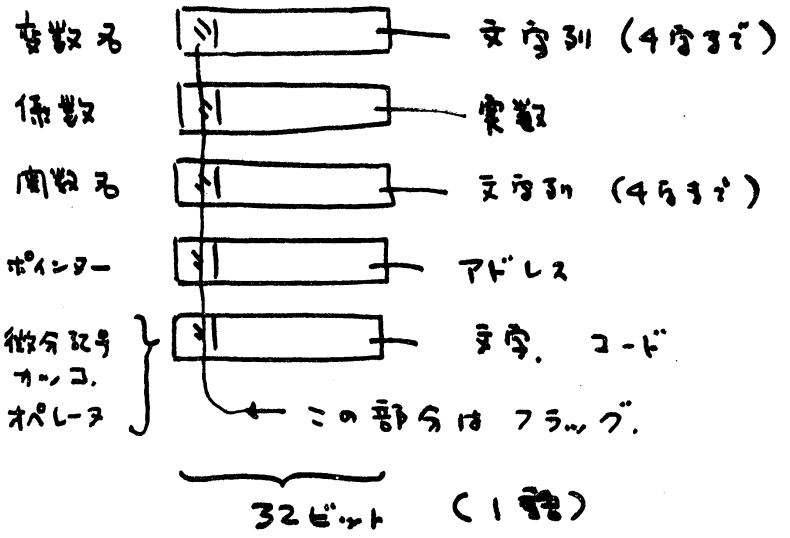
} ここは、まだ安全に動いていないが。

- ☆ (ドラムにある式をコアの式に)代入する。
- ☆ せいとん、単純化せよ。
- ☆ 微分を実行せよ。
- ☆ 展開せよ。
- ☆ プリントせよ。

[入力フォーマット]　FORTRAN の算術式と全く同じ.[注2]

[出力フォーマット]　〃

[内部表現]　ここが大変. 風変りで. (そのために損をしているところで). 外部表現をほとんどそのまま使っている ——— 木構造になっていない ——— そのためにディバッグの入出力は簡単でよいが. 処理自体は. かなり困難であった (論理の見落としが多い)。



このようにすると.

MOVE は簡単.

メモリーの使用効率は. すこぶる良い.

式の処理のロジックは. すこぶる複雑

になる。係数の整数処理ということは. 考えていなかった (それはあまり いいことではない)。

注2). ただし変数名は4文字までで添字付は許さない。

[処理プログラムの記述言語] FORTRAN. (HITAC 5020用)

unpack の部分も最初は FORTRAN で書いたが. あとでアセンブラのを作った (しかし他の処理に含まれる時間が多かったので. 時間的にはあまり変らなかった)

記述言語としては. ——— やりたいことを表現するという意味では ——— FORTRAN で特に困ったことはないが. その通りにコンピュータが動くかどうかということはまたべつで. また実行効率という点では. どうもロスが多すぎる.

(例). あるデータの. A というフィールドを用いるとき.

AFIELD (DATA) ..... 関数

と書けば. 表現はできるが. それを何ヶ所にも使えば時間が掛かるし. 置換えをやめては. 複雑な nesting になっている部分では非常に危険である.

(例). モード・インディケータのようなものが. サブルーチンのリンクの際に渡しにくい. COMMON を使って. それもラベルを用いても. かなり多数の情報を受渡すので大変. PL/I のようになっていると有難い.

そのかわり係数の計算などは簡単.

[単純化] ほかの処理との関連で、能率がいろいろ変ってくる。たとえば積を作るとき、あらかじめすべての項をABC順（重数なの）に並べておいて、積の計算の時には単につなぐだけでなくマージしてしまうとよいが、常に標準型を保つということの為の仕事がふえる。

微分しながら、係数が0と1になるのをチェックすることもできるが、「そこでチェックすれば、ほかで単純化しないでいい」ということにはなるべくムダにすることもある。

サブルーチンに入る時、「オペランドとしての式は単純化されている」という仮定があると便利な場合が、これを堅持することは、かなり困難で、ことに機能を追設する場合にトラブルが起こる。サブルーチンの入口で標準型かどうかチェックするのも時間的ロスが大きい。しかし、「どんな式でも」というため内部処理を複雑にするのと、どちらがいいか、少々、問題がある。

e^0、x^0などは1にする。0/0が出てくるとき、「0がかかっているから答は0にする」という操作を先にやってしまうと、おかしな答が出る。

[思うこと] いろいろチェックをしてみると、実用的な問題がたいてい通っても、人工的にトリビアルな問題を作って入れてみると、よく脱線することがわかった。「数式解析のロジックが正しい」という証明をするプログラム」が必要であると感じた。

[文献]。

第9回、プログラミング・シンポジウム 報告集。

構造記号情報研究会。微刷、(1968 ?)

[補足]。

代入の逆のロジックは、可能か？

例) $z=(ax+b)\cdot e^{(ax+b)}$

$$\longrightarrow \begin{cases} y = ax+b \\ z = y e^{y} \end{cases}$$

Σ の演算の問題点。

$$cx + \sum_{n=2}^{m} a_n x^n \longrightarrow \sum_{n=1}^{m} a_n x^n$$

$$\sum_{k=1}^{m} a_k + \sum_{n=1}^{m} b_n \longrightarrow \sum_{i=1}^{m} d_i$$

絶対値がつくと、「条件付の式」になる。例、$\frac{d|x|}{dx} =$