電子計算機による exact な計算の新方法*

(modulo p 演算の応用)

高橋秀俊** 石橋善弘**

1. 序論

電子計算機による計算の精度，つまり桁数は，それぞれの計算機によって定っているのが普通である（たとえば2進36桁とか10進10桁のように）．それより桁数の多い結果がたとえば乗算によって出て来た場合は，下の方の桁は“丸め”によって除かれてしまう．しかし，時にはそのような丸めを全く行わない厳密（exact）な計算が必要になることがある．その場合は普通は，桁数の多い数をいくつかに分割して，いわゆる多重精度のサブルーチンなどを使って計算することになる．ここでのべるのはそのような多重精度の計算を，簡単化する新方法である．

exact な計算は整数論に関係した問題，たとえば素数の計算，素因子分解，不定方程式の解などで必要になってくる．しかし，必らずしもこの種の問題に限られるわけではない．数学解析の問題のうちでも，たとえば行列の逆転などで，しばしば“たちの悪い”（ill-behaved な）場合があらわれる．行列の行列式の計算の際二つの大きい数の減算で頭の方の桁が大きく消えてしまって有効数字が減るような場合である．この種の問題には，浮動小数点はあまり助けにならないばかりか，このような精度低下を蔽い隠す危険をもっている．この種の場合は，多重精度演算によって精度を上げるほかないが，最も徹底したやり方は，最初に与えられたデータを整数と見て，最後まで丸めを行わずに厳密に計算を実行することである．

行列式の計算のような型の計算を，厳密に遂行するには若干の困難を伴う．行列式を求めるのに，展開式によって項別に計算することは，行列の次数が少し多くなると実際上不可能になるので，適当な消去法（掃き出し法）によるべきであるが，掃き出し法である列の要素を一つを除いて消してしまうには，いわゆる軸（pivot）にした要素の値を各行に乗じなければならず，行列式の値には，そのため，大きい乗数が掛けられることになる．

このようなことを繰り返して行くと行列の要素の値は，掃き出しの度毎に急速に増して行くであろう（一回の掃き出し毎に2倍の長さになる）．そうして最後に得た値を，それまでに乗じた数で割れば（必ず割り切れて）正しい答えが出てくる．しかし，このような方法では，最後の必要な答の大きさよりはるかに大きい整数を途中で扱わなければならず，実際的でない．

一般に計算の内容によって，最後の答よりはるかに大きい数が中間的に出てくるような場合がほかにもしばしばあると思われる．

このような，整数の加減算と乗算だけから成り立つ長大な計算（除算が含まれれば，答を分数で出すことにすればよい）を著るしく簡単に行う方法がここにのべるものである．

この新方法の要点は，計算を一つの**整数 p を法（modulus）として**計算すること，つまり，p より大きい整数が出たらこれを p で割った剰余をそのかわりに採用して計算を進めるということである．そうすれば，p がその計算機の扱い得る範囲の桁数の数であるならば，計算はどこまでも単一精度の計算として進められるわけである．

もちろん，こうして得た答は，真の値そのものではなく，真の値を p で割った剰余である．しかし，これから真の値を逆に求めることができる．それには，同じデータによる同じ計算をいくつかの異る法 $p_1, p_2, \cdots p_l$ について行い，そうして得た答 $u_1\ u_2, \cdots u_l$ に，それぞれの法に関して相合になるような大きい整数 u を決定するのである．一般に，もとの計算式から，u の取り得る値の上限を評価することは可能だから，その上限の大きさから，何個の p を使えば u を一義的に定められるかがわかる．

上にも注意したように，この方法で扱い得るのは，本質的には加減乗の三種の演算だけである．しかし例外として，除法が含まれた計算であっても，結果が最

---

* A New Method for the “Exact Calculation” by a Digital Computer, by Hidetoshi Takahasi and Yoshihiro Ishibashi (Faculty of Science, University of Tokyo)

** 東京大学理学部

終的には割り切れて，整数の答が出るということがはじめからわかっている場合（たとえば$\frac{n(n-1)(n-2)}{6}$を計算するというような場合）にかぎり，われわれの方法が適用できるのである．すなわちその場合は，実際に modulo p による除算を行う．つまり $ax\equiv b(\mathrm{mod}\ p)$ を解くことによって得た x を以て b/a をあらわすことにすればよいのである．なお“除算”が正当化されるためには p が素数であることが要求される．

この方法の原理は“casting out nines”と呼ばれるよく知られた検算法と同じであることに注意しておく．たとえば 1234+567 という演算は，

$$\left.\begin{array}{l} 1234 \equiv 1+2+3+4\equiv 10\equiv 1+0=1 \\ \underline{\ 567 \equiv\ \ 5+6+7\equiv 18\equiv 1+8\equiv 9\equiv 0} \\ 1801 \equiv 1+8+0+1\equiv 10\equiv 1+0\equiv 1 \end{array}\right\} (\mathrm{mod}\ 9)$$

$$1+0\equiv 1\ (\mathrm{mod}\ 9)$$

のようにして検算される．これは正にわれわれの方法で $p=9$ にした場合である．一般に

$$A\equiv A_0,\ B\equiv B_0\ (\mathrm{mod}\ p)$$

ならば

$$\left.\begin{array}{l} A\pm B\equiv A_0\pm B_0 \\ AB\equiv A_0B_0 \end{array}\right\} (\mathrm{mod}\ p)$$

である．という定理によって，$A\pm B$, AB を計算するかわりに $A_0\pm B_0$ あるいは A_0B_0 を計算するというのが本方法の原理である．

2. 計算方法

2・1 modulo p での簡約

以上のべた原理ですべては明白で，ほかには，もはやほとんどいう必要がないのであるが，この方法をプログラムによって実行する際に問題となることについて少しのべよう．

この計算には，modulo p による加減算，乗算，および必要なら反転即ち，a が与えられたとき

$$ax\equiv 1\ (\mathrm{mod}\ p) \qquad (2\cdot1\cdot1)$$

をみたす x を求める計算を行うサブルーチンが必要である．むしろこれらをまとめ，若干の判断命令を附加して，浮動小数点ルーチンで行われるのと同様に解釈ルーチン（interpretive routine）としておくと便利である．

これらのサブルーチンは，mod p で簡約された数，つまり $0\leqq x<p$ となるような x を operand として演算し，その結果をやはり上の範囲に入るように標準化して出すようにつくる必要がある．任意の整数 X を mod p で簡約するとは

$$X=pQ+x\quad 0\leqq x<p \qquad (2\cdot1\cdot2)$$

となるような Q, x の組を見出すことである．これは計算機に正しい剰余が出る除算命令が用意されていれば，X を（整数として）p で割って余りを x とするだけであるから簡単である．しかし，p をその計算機の演算装置の扱う最大数よりわずか小さい値にしておくと，次のような方法が行われて便利である．計算機がたとえば2進 S 桁したとき

$$p=2^S-h \qquad (2\cdot1\cdot3)$$

とおく．h は小さい整数とする．そこで X を p で割った余りを求めることは，X の overflow した部分，つまり $X/2^S$ の整数部 X_u を h 倍して，これを X の overflow せずに残った部分 X_l に加えることである．即ち

$$X=2^SX_u+X_l,\ 0\leqq X_l<2^S \qquad (2\cdot1\cdot4)$$

であるから

$$X'=X_l+hX_u=X_l+(2^S-p)X_u$$
$$=X-pX_u\equiv X\ (\mathrm{mod}\ p) \qquad (2\cdot1\cdot5)$$

だからである．もし，こうして出た X' がまた overflow している場合はもう一度同じ操作を行う．こうして得た結果が，$0\leqq X'<p$ となっていれば簡約ができたことになる．この方法は正しい除算のできる命令のない機械にも向いている（これは実は casting out nines 法そのものである）．この簡約法で具合の悪いのは

$$p\leqq X'<2^S \qquad (2\cdot1\cdot6)$$

になった場合である．これは overflow していないから，上の方法ではこれ以上どうにもならないことになる．この困難を避けるため，われわれのサブルーチンでは X を最初負の値になるようにする．そうして簡約を行えば，$X_u<0$ であるから決して正に overflow しない．そこで簡約操作をつづけるうち $X_u=0$ となったら終りにして X_l を取り出すのである．こうすれば必ず正しい範囲に簡約される．

2・2 逆数の計算

a が与えられたとき方程式

$$ax\equiv 1\ (\mathrm{mod}\ p) \qquad (2\cdot2\cdot1)$$

を解く方法には Euclid の互除法によるものと，Fermat の定理を使うものとが知られている．恐らく前者の方が計算量は少いが，後者の方がプログラムしやすいので，後者を用いることにする．p が素数ならば

$$a^{p-1}\equiv 1\ (\mathrm{mod}\ p) \qquad (2\cdot2\cdot2)$$

であることから

$$x\equiv a^{p-2}\ (\mathrm{mod}\ p) \qquad (2\cdot2\cdot3)$$

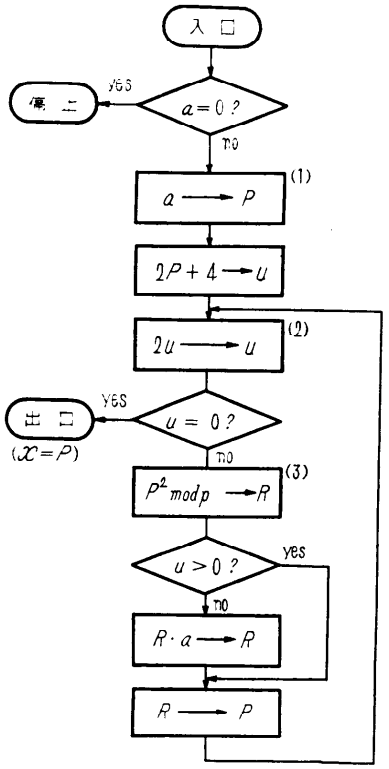
によって x を求めるのであり，$p-2$ が2進法で

$$p-2=\sum_{i=0}^{s-1} d_i z^i \qquad (2\cdot2\cdot4)$$

の形をもつとすると，漸化式

$$\left.\begin{aligned} &P_0=1\\ &P_i\equiv P_{i-1}{}^2\cdot a^{d_{s-i}} \\ &x\equiv P_s \end{aligned}\right\} \ (\mathrm{mod}\ p)\ (i=1,2,\cdots s) \qquad (2\cdot2\cdot5)$$

によって x がきまる．即ち $p-2$ の 2 進表示を bit ごとにしらべて，それによって $P^2{}_{i-1}$ に a を乗ずるか乗じないかを判断するようなプログラムでできる．このプロセスは 2 進法の計算機に特に適している．**第1図**に，この方法による逆数ルーチンの流れ図を示す．



第 1 図　mod p 逆数ルーチンの流れ図

〔註〕(1) $2^{S-1}+2<P<2^S$ とする．したがって d_{S-1} は常に 1 だから最初の 1 step は loop の外で行った．
(2) u は $P-2$ の各 digit を見るための strobe．これを左へシフトしながら sign digit をしらべる．
(3) R は PC-1 の R-レジスタ

10進法の計算機でも，p の 2 進表示をつくりながら上と同じプロセスを行うのが便利と思われる．

2・3 modulo p_i による計算値から真値を求めること

以上の方法で，ある同じ計算を幾つかの法 $p_1, p_2, \cdots p_n$ に対して行って，

$$Y\equiv y_i(\mathrm{mod}\ p_i)\ (i=1,2,\cdots n) \qquad (2\cdot3\cdot1)$$

を得たとする．真の値 Y がどの p_i よりも小さければ y_i はすべて等しく，それが Y そのものを与えるわけであるが，一般には，Yは計算機の演算容量を超える大きさになり（その場合こそこの方法が有効なのであるが），$y_1\ y_2,\ y_3\cdots$ はみな異った値となる．そのとき，(2・3・1) を連立方程式として解いて Y を一義的に決定することが問題になる．

Y が一義的に決定できるためには，Y に対するある条件が必要である．たとえば

$$0\leqslant Y<p_1p_2\cdots p_n \qquad (2\cdot3\cdot2)$$

あるいは $\qquad |Y|<1/2p_1p_2\cdots p_n \qquad (2\cdot3\cdot2')$

のような条件である．一般に計算式の性質から $|Y|$ の取り得る上限はわかるから n を幾つにとれば (2・3・2) がみたされるかが決定できる．そのように n が選ばれたとして，Y を求めるため

$$\begin{aligned} &Y\equiv Y_1(=y_1)\ (\mathrm{mod}\ p_1)\\ &Y\equiv Y_2 \quad (\mathrm{mod}\ p_1p_2)\\ &Y\equiv Y_3 \quad (\mathrm{mod}\ p_1p_2p_3)\\ &\quad\vdots \qquad\qquad \vdots\\ &Y\equiv Y_i \quad (\mathrm{mod}\ p_1p_2\cdots p_i) \end{aligned} \qquad (2\cdot3\cdot3)$$

のような $Y_1, Y_2, Y_3, \cdots, Y_n$ を逐次定めることを考える．それは

$$\left.\begin{aligned} &Y_2-Y_1=q_2p_1\\ &Y_3-Y_2=q_3p_1p_2\\ &\quad\vdots \qquad\quad \vdots\\ &Y_i-Y_{i-1}=q_ip_1p_2\cdots p_{i-1} \end{aligned}\right\} \qquad (2\cdot3\cdot4)$$

のように置くことによって得られるはずである．即ち $Y-Y_{i-1}$, $Y-Y_i$ はともに $p_1p_2\cdots p_{i-1}$ の倍数であるから，その差を(2・3・4)の右辺のように置くのである．ここに一般に

$$0\leqslant q_i<p_{i-1} \qquad (2\cdot3\cdot5)$$

とすることができる．そこで q_i は

$$Y_i-Y_{i-1}\equiv y_i-Y_{i-1}\ (\mathrm{mod}\ p_i) \qquad (2\cdot3\cdot6)$$

によって

$$q_i\equiv(y_i-Y_{i-1})(p_1p_2\cdots p_{i-1})^{-1}\ (\mathrm{mod}\ p_i) \qquad (2\cdot3\cdot7)$$

からきまる．ここで $(\ )^{-1}$ はもちろん mod p_i に関する逆数を意味する．

こうして求めた最後の Y_n が Y に等しいわけである．あるいは (2・3・4) をすべての i について辺々加えて書きなおせば

$$Y=(((\cdots(q_np_{n-1}+q_{n-1})p_{n-2}+q_{n-2})p_{n-2}+\cdots)\cdot p_1+q_1 \qquad (2\cdot3\cdot8)$$

と書ける．この $q_1(=y_1), q_2, q_3, \cdots q_n$ を (2・3・7) によって求めればよいのである．

即ち，(2·3·4)，(2·3·7) によって $Y_2, Y_3, Y_4, \cdots$ を次々に求めて行き最後の Y_n を以て答とすればよい．

Y の範囲が (2·3·2) によって与えられるときは，q_i は (2·3·5) によってきめればよいが，Y の範囲が (2·3·2′) で与えられる場合は，

$$|q_i| < p_{i-1}/2 \qquad (2\cdot3\cdot9)$$

となるように q_i を決定する方がよい．いずれの場合も，必要以上に大きい n を取った場合は，q_i があるところから先は0になる．逆に，ある q_i が0になったら，Y_{i-1} が正しい Y であることはほとんど確実であるといえる．しかし，あとから0でない q_i が出てくることも不可能ではないから，確実に $Y_{i-1}=Y$ である保証はない．

以上の計算に必要なのは，多重精度の数の加算，多重精度の数と単一精度の数との乗算，および多重精度の数の modulo p による簡約である．なお，$(p_1p_2\cdots p_{i-1})^{-1}$ (mod p_i) はあらかじめ計算しておけばよい．その表を**第 1 表**に示す．

第 1 表

i	p_i	h_i	$(p_1p_2\cdots p_{i-1})^{-1} \bmod p_i$
1	3 43597 38337	31	1
2	38319	49	3 24508 63968
3	38307	61	73491 66249
4	38299	69	2 24977 16804
5	38289	79	2 11820 09803
6	38247	121	3 34384 41941
7	38227	141	1 79709 97103
8	38121	247	2 18289 14109
9	38059	309	2 72955 85313
10	38043	325	68501 32122
11	38011	357	1 04016 19371
12	37917	451	1 46250 41267
13	37869	499	5351 31435
14	37849	519	1 26517 94573
15	37837	531	2 06522 59502
16	37821	547	1 74572 11185
17	37813	555	1 40672 61446
18	37791	577	1 85955 21066
19	37777	591	1 01995 05182
20	37771	597	1 46932 64241

ただし $h_i = 2^{35} - p_i$

2·4 本方法による計算量

modulo p による本方法と，普通の多重精度計算とで同じ問題（加減乗算だけでできるもの）を行った場合の計算の量を比較して見る．n 重精度が必要な計算とすると，本方法では p_i はその計算機の容量に一ばいの大きい素数をとることにすれば，n 個使えば十分と考えられるので，その仮定で比較すると，n 重精度の数同志の乗算には n^2 回の乗算が必要であるが，modulo p の場合は modulo p の乗算を n 回行えばよい．そのかわり modulo p で簡約するのに乗算が平均1回必要となるから，むしろ乗算回数は $2n$ という方がよいかもしれないが，それにしても n が大きいときは，後者の方がはるかに有利なことがわかる．もう一つの利点は，法 p をいろいろ変えて行う計算はそれぞれ全く別に行えばよいから，演算に要する記憶場所は単一精度の場合と同じでよいことで，記憶容量の制限された計算機に有利である．

3. 応　　用

この方法の最も効果の顕著な応用例として，行列の反転の問題を考えよう．逆転すべき行列の要素は整数で与えられてあり，逆行列は分数の形で，分母にその行列の行列式を，分子にその小行列式を書いた形で求めるものとする．

これに用いる方法は，普通の消去法そのものである．与えられた方程式を

$$\begin{pmatrix} a_{11} & a_{12}\cdots a_{1n} \\ a_{21} & a_{22}\cdots a_{2n} \\ \vdots & \vdots \\ a_{n1} & a_{n2}\cdots a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} 1 & 0\cdots 0 \\ 0 & 1\cdots 0 \\ \vdots & \vdots \\ 0 & 0\cdots 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \qquad (3\cdot1)$$

とする．これの第1行を a_{11} で除して

$$\begin{pmatrix} 1 & a_{12}/a_{11}\cdots a_{1n}/a_{11} \\ a_{21} & a_{22}\cdots a_{2n} \\ \vdots & \vdots \\ a_{n1} & a_{n2}\cdots a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} 1/a_{11} & 0\cdots 0 \\ 0 & 1\cdots 0 \\ \vdots & \vdots \\ 0 & 0\cdots 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \qquad (3\cdot2)$$

これから

$$a_{ij}' = a_{ij} - \frac{a_{1j}}{a_{11}} a_{i1} \qquad (3\cdot3)$$

$$b_{ij}' = b_{ij} - b_{1j}a_{i1} \qquad (3\cdot4)$$

(b_{ij} は右辺の行列の成分)

によって方程式を変換して，左辺の第1列を消去すると

$$\begin{pmatrix} 1 & a_{12}/a_{11}\cdots a_{1n}/a_{11} \\ 0 & a_{22}' \quad a_{2n}' \\ \vdots & \\ 0 & a_{n2}' \quad a_{nn}' \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} 1/a_{11} & 0\cdots 0 \\ -a_{21}/a_{11} & 1\cdots 0 \\ \vdots & \\ -a_{n1}/a_{11} & 0\cdots 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \qquad (3\cdot5)$$

次に，同様にして a_{22}' を軸として“掃き出し”操作を行って左辺第2列を対角線上を除いて消去する．このようにして，行列を全部逆転するのがよく知られた行列逆転法である．

さて，この方法を exact に遂行しようとすると，分数の加減算のとき毎回分母を掛け合わせて共通分母に

しなければならないので，分数の分母分子が非常に大きくなる．しかし，われわれの方法では，どんなに大きい数になっても modulo p で簡約してしまうから，べつに支障にはならないので，そのような方法ももちろん実行可能である．

しかし，ここでは modulo p の計算の特質をもっと徹底的に活用した方法を用いる．それは，消去法の際でてくる除算を全部 modulo p で実行してしまうのである．たとえば，7で割る計算があれば

$$7^{-1}\equiv 29451204289\ (\mathrm{mod}(2^{35}-31))$$

としてしまう．

$$7\times 29451204289=206158430023\equiv 1(\mathrm{mod}(2^{35}-31))$$

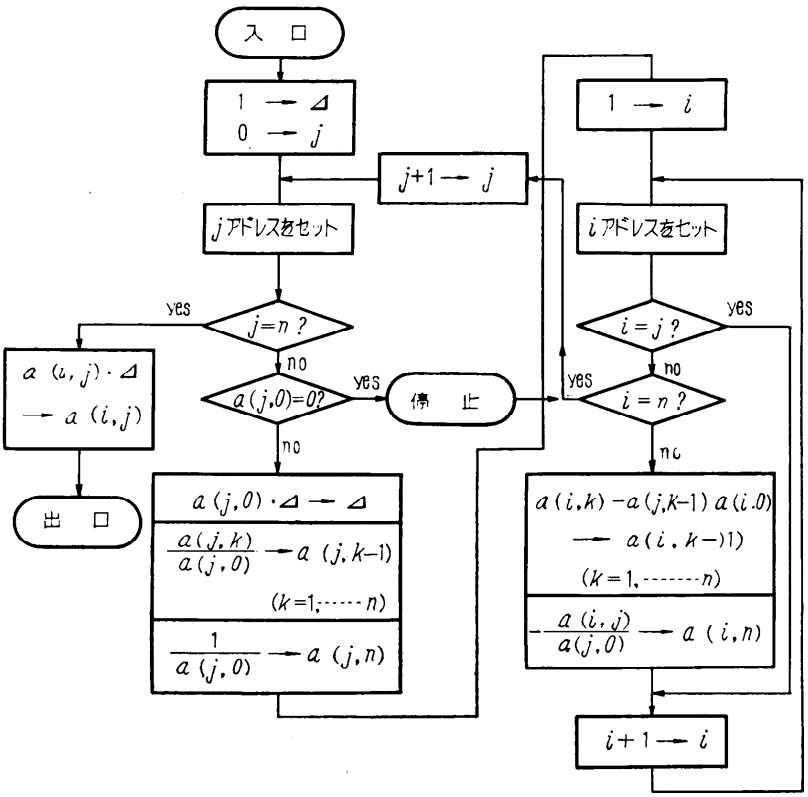
だからである．このようにすると，もはや(3・5)のような行列に出てくる要素の値は，一見何だかわからない数字の羅列になってしまうが，実はそれらはそれぞれある分数をあらわしているのである．

そのようにして modulo p での除算をかまわず実行しながら，前述のような消去プロセスをつづけて行くと，終いに左辺は単位行列になり，右辺に逆行列ができ上る．ここで，この右辺はそのままでは意味のわからないものである．即ち，各要素は一つの分数をあらわしているが，その分数は何であるかは，これだけから決定することは困難である．しかし，一方，一次方程式の理論によれば，整数を要素とする行列の逆行列の要素は，その行列式を分母とした一つの分数であらわされることがわかる．ということは，計算で出たこれらの行列要素のおのおのに，一つの整数 Δ を乗ずると，全部が本当の整数になるということである．この整数とわかった数の値は，上にのべた方法で，いくつかの法に対する剰余値から一義的に決定できる．即ち，逆行列の各要素の分子が決定される．分母はすべてを通じ Δ である．

行列式 Δ を計算することは，掃き出しの操作の中で簡単にできる．消去演算のとき出てくる“軸”の値を $a_{11}, a_{22}', a_{33}'', \cdots a_{nn}^{(n-1)}$ とするとき，これら全部の積

$$\Delta=a_{11}a_{22}'a_{33}''\cdots a_{nn}^{(n-1)} \qquad (3\cdot 6)$$

が行列式の値にほかならないのである．なぜなら，毎回の掃き出しプロセスの中で，左辺の行列の行列式が変化するのは，最初に軸を含む行の各要素を a_{11}（または a_{22}', a_{33}''，等）で割る段階であって，その後の掃



第 2 図　行列反転ルーチンの流れ図

き出し操作は行列式の値を変えない．そこで，行列式の真の値を Δ とすると，第 1 回の掃き出しで Δ/a_{11} 第 2 回の掃き出しで $\Delta/(a_{11}a_{22}')$,… となり，最後に得られる左辺の対角行列の行列式の値は

$$\Delta/(a_{11}a_{22}'a_{33}''\cdots a_{nn}^{(n-1)}) \qquad (3\cdot7)$$

となる．ところが，この行列式の値は明らかに 1 であるから，$a_{11}\ a_{22}'\ a_{33}''\cdots a_{nn}^{(n-1)}$ が Δ に等しいのである．

以上のことを考えて，つくった行列反転プログラムの流れ図を**第 2 図**に示す．

ここで面倒なのは，軸とすべき要素 $a_{ii}^{(i-1)}$ がちょうど 0 になる場合である．その場合は，その下の要素 $a_{i+1,i}^{(i-1)}$, $a_{i+2,i}^{(i-1)}$… を順にしらべて，0 にならないものにあたったら，それを軸として消去する．しかし，そのようなことがあると，どの行についてはすでに掃き出しが終っているかを何かの方法で印をしなければならない．われわれの mod p のプログラムでは，数値の符号の桁は使用されていない（数値はすべて正の値に規格化されている）ので，掃き出しを行う度毎に，軸となった行の要素全部（軸要素を除く）に符号の桁を 1 にしてマークするような方法が考えられる．しかし，0 の要素が出ることは比較的まれであるから，その場合は計算を止めて，行または列を入れかえたものについて計算をやりなおすことでも足りるであろう．

4. 多項式に対する演算の新方法

4·1 — 般　論

今までのべた方法の根本思想を少し範囲をひろげて考えるのも興味がある．たとえば modulo p のかわりに，modulo $f(x)$ を考えるのである．ここに $f(x)$ は x の n 次多項式であって，その係数は整数であっても，また任意の実数であっても，あるいは複素数であってもよい．ところで，任意の多項式 $P(x)$ の modulo $f(x)$ による剰余を考えることは

$$f(x)=0 \qquad (4\cdot1\cdot1)$$

の根を $\alpha_1, \alpha_2,\cdots\alpha_n$ としたとき，$P(\alpha_1)$, $P(\alpha_2)$,…$P(\alpha_n)$ を考えることである．即ち，上の思想を，多項式に対する演算に応用すれば，多項式の加減乗算を式の形で直接行うかわりに，変数 x に $\alpha_1, \alpha_2,\cdots\alpha_n$ などの数値を次々に代入して，それぞれの場合の sample 値 $P(\alpha_1)$, $P(\alpha_2)$,…$P(\alpha_n)$ を求め，それから，逆に

$$P(x)=\sum_{i=0}^{n-1}a_ix^i \qquad (4\cdot1\cdot2)$$

の各項の係数 a_i を定めるという考えに到達する．ここに $P(x)$ が $n-1$ 次以下であれば（そうして，$\alpha_1\cdots\alpha_n$ はみな異っていれば），いつも多項式 $P(x)$ の形，つまり a_i は一義的にきめられる．それには内挿公式，たとえば Lagrange の内挿式を使えばよいわけである．

この方法の利点は，前の場合と全く同じで，一般に多項式の演算をプログラムすることは複雑であり，しかも多数の記憶場所を必要とするのに対して，上のように，多項式の特定の x に対する sample 値についての演算だけならば普通の演算命令を使った直接プログラムが使われ，記憶場所も少くてすむ．そうして，最後に，得られたいくつかの sample 値から多項式を決定するプログラムを使って，係数を決定すればよい．この最後の段階は標準的なプログラムを用意しておけばよいから問題はないので，個々の問題のために組まねばならないプログラムが著しく簡単になる点が重要である．

この場合，何個の α を取らねばならないかは，答が何次の多項式になるかによるわけであるが，一般に多項式に関する有限な演算では，答が何次式になるかは予定できるのが普通であるから，それに必要なだけの α を使って計算すればよい．

この方法の最も有効な例は，交流回路の計算であろう．交流回路の計算では一般に多項式

$$L_{ik}p^2+R_{ik}p+D_{ik} \qquad (4\cdot1\cdot3)$$

を要素とする行列式，あるいは行列の計算が必要となる．ここで必要とされるのは，終局的には数値である場合が多いのであるが，それは通常 p の複素値（主として純虚数値）に対する値であって，その計算には，複素数演算を行わなければならない．ところが，ここでのべた方法によると，まずこの p に実数の値 α_1, $\alpha_2,\cdots\alpha_n$ を入れて，それぞれに対する行列式その他の数値を計算し，それから，それらをあらわす多項式の係数を決定し，最後に，必要があれば，p に虚数値を入れて，複素数演算を行うのである．この場合，複素数演算は多項式の計算という定った形になおした後で行えばよいのだから，標準化されたルーチンを用意しておくことが容易であり，最も複雑な行列計算の部分を，実数値に対して行えばよいというのが重要な利点なのである．今日，たとえば 10 行 10 列の行列式で，その各要素が p の 2 次多項式であるものを展開す

る方法として，実行可能なものは恐らく知られていないと思われる．この方法は，その場合の唯一の方法であると断言してもよいと思われる．

この方法を適用する場合，計算のもとになる要素多項式がすべて整数係数であれば，この場合の数値計算に更に，前述の modulo p の計算を使うことができる．そうすれば，得られる sample 値は modulo p で正しく，したがって，いくつかの p について計算することによって多項式の exact な sample 値が求められ，それから多項式の係数の値が exact に求められる．exact な計算にすることの利点は，sample として採用する α として何を取るかによって結果の精度が左右されないことである．たとえば簡単に，0, 1,2…, $n-1$ と取って，そこでの $P(x)$ の値 $P(0)$, $P(1)$, $P(2)$,… $P(n-1)$ を計算すれば，それから $P(x)$ を正確に決定できるという点である．exact でない場合は，内挿式を外挿に使ったり，実数値に対して sample してきめた多項式に複素数を代入したりしたときには精度が低下するおそれがあるのである．

4・2 多項式の決定法について

上に，たとえば Lagrange の内挿式を使えばよいとのべたが，実はそのような式を使うのはあまり便利とはいえない．むしろ，直接に

$$\sum_{i=0}^{n-1} a_i \alpha_j{}^i = P(\alpha_j) \quad j=1,2,\cdots n \qquad (4\cdot2\cdot1)$$

という a_i に対する連立方程式を普通の方法で（あるいは本文でのべた方法で）解くか，そうでなければ，modulo p の計算のときにのべた方法にならって，次のようにする．$P(x)$ を求めるために，多項式の系列 P_0, $P_1(x)$, $P_2(x)$,…$P_{n-1}(x)=P(x)$ を，P_0 は $x=\alpha_1$ で $P(x)$ に一致，$P_1(x)$ は $x=\alpha_1$, α_2 で $P(x)$ に一致，$P_2(x)$ は $x=\alpha_1$, α_2, α_3 で $P(x)$ に一致…というように，順により高い一致が得られるようにする．それは

$$\left.\begin{aligned}
&P_0=P(\alpha_1)\\
&P_1(x)=P_0+(P(\alpha_2)-P_0)\frac{x-\alpha_1}{\alpha_2-\alpha_1}\\
&P_2(x)=P_1(x)+(P(\alpha_3)-P_1(\alpha_3))\\
&\qquad\cdot\frac{(x-\alpha_1)(x-\alpha_2)}{(\alpha_3-\alpha_1)(\alpha_3-\alpha_2)}\\
&P_i(x)=P_{i-1}(x)+(P(\alpha_{i+1})-P_{i-1}(\alpha_{i+1}))\\
&\qquad\cdot\frac{(x-\alpha_1)(x-\alpha_2)\cdots(x-\alpha_i)}{(\alpha_{i+1}-\alpha_1)(\alpha_{i+1}-\alpha_2)\cdots(\alpha_{i+1}-\alpha_i)}
\end{aligned}\right\} \qquad (4\cdot2\cdot2)$$

であらわされる．ここで P_1, P_2, P_3… と多項式を順繰りに計算して行く手続きはかなり簡単である．

modulo p を使って exact に計算する場合には，$\alpha_1=0$, $\alpha_2=1$,…$\alpha_n=n-1$ と置けばよいのであるが，そのとき右辺の第2項にあらわれる $1/i!$ という因数に相当する割算は，この場合 modulo p での $i!$ の逆数を使用してかまわない．そうしても分子の方の係数が（真の値は）必らず $i!$ で割り切れる数になっているので，結果としては正しい係数値が出るのである．

4・3 整数論的調和解析

この方法で用いる sample 値 α_i としては何をとってもよいから，係数 a_i の決定が特に簡単になるように適当なものを選ぶことが考えられる．最もよいと思われるのは，ε を方程式

$$\varepsilon^n \equiv 1 \pmod{p} \qquad (4\cdot3\cdot1)$$

の原始根（つまり ε, ε^2, ε^3,…ε^{n-1} のうちどれも1に等しくならないような根）とするとき

$$\alpha_i \equiv \varepsilon^i \pmod{p} \qquad (4\cdot3\cdot2)$$

とすることである．即ち

$$P(\varepsilon^j)=\sum_i a_i \varepsilon^{ij} \qquad (4\cdot3\cdot3)$$

を計算するのである．この式をみると，$P(\varepsilon^j)$ は a_i に対する mod p に関する Fourier 変換というべきものであることがわかる．そこで，a_i は $P(\varepsilon_i)$ から Fourier 逆変換の式

$$a_i=n^{-1}\sum_j \varepsilon^{-ij}P(\varepsilon^j) \qquad (4\cdot3\cdot4)$$

によって求められる．いわば整数論的調和解析である．ここで n は $P(x)$ の次数より大きくなければならないのはもちろんである．

ここで(4・3・1)が整数の原始根を有すること，つまり x^n-1 が mod p で一次因数に分解されることの必要かつ十分な条件は，n が $p-1$ の約数であることである．その場合，u を mod p に対する1の原始根（原始 $p-1$ 乗根）とすると，

$$\varepsilon \equiv u^{(p-1)/n} \pmod{p} \qquad (4\cdot3\cdot5)$$

によって ε が与えられる．ε の選び方は幾とおりもあるが，どれを使っても最後の結果（a_i の値）は同じになる．

4・4 複素数計算

調和解析の方法で $n=4$ とした場合は，実は複素数つまり $a+b\sqrt{-1}$ の形の数に対する計算法を与える．即ち，この場合は ε は

$$\varepsilon^2 \equiv -1 \pmod{p} \qquad (4\cdot4\cdot1)$$

の根であり，与えられた計算式の中の $\sqrt{-1}$ のかわりに全部 ε を代入して計算した結果を $P(\varepsilon)$，また $-\varepsilon$ を代入して計算した結果を $P(-\varepsilon)$ とすると，実

際の結果 $A+B\sqrt{-1}$ は

$$\left.\begin{aligned} A&=2^{-1}(P(\varepsilon)+P(-\varepsilon)) \\ B&=2^{-1}(P(\varepsilon)-P(-\varepsilon)) \end{aligned}\right\} \qquad (4\cdot4\cdot2)$$

によって与えられる.

5. 数 値 例

例 1.

$$\Delta=\begin{vmatrix} x-11 & -41 & 26 & 19 & -72 & 93 & 32 & 18 & 70 \\ -89 & x-11 & -4 & 88 & -33 & -87 & -53 & -89 & -67 \\ 88 & 49 & x+45 & -71 & -63 & -82 & -85 & -71 & 6 \\ 55 & -93 & -55 & x+92 & 5 & 86 & 76 & -84 & 12 \\ -21 & 84 & -38 & -18 & x+88 & 44 & -33 & 51 & -96 \\ 34 & -20 & -95 & 4 & -51 & x+34 & -92 & 2 & -71 \\ 17 & -40 & 44 & -87 & -31 & 37 & x-89 & 69 & -21 \\ 47 & -25 & 85 & 25 & -25 & -51 & -30 & x+11 & 31 \\ -68 & -81 & 7 & 63 & 67 & -70 & 25 & 43 & x-5 \end{vmatrix}$$

を計算すること.

これは9次式であるから, x を 1 の n 乗根とすると n は少なくとも 10 またはそれ以上でなければならない. たとえば $n=11$ にとってみる.

$$x^{11}\equiv1 \pmod{p} \qquad (5\cdot1)$$

が分解できるのは, $p-1$ が 11 の倍数となる場合で, $p=2^{35}-31$, $2^{35}-141$ の二つは条件に合う. Δ の値は p を2種取れば足りる程度の大きさである. (5·1) の解として

$$x_1\equiv 2284082430 \pmod{(2^{35}-31)}$$
$$x_1\equiv 12062584333 \pmod{(2^{35}-141)}$$

を得る (他の根は, $x_r=x_1^r$, $r=1,2,3,\cdots n$ で与えられる) これらを入れて, Δ を計算すると

	$p=2^{35}-31$	$p=2^{35}-141$
$\Delta(x_1)\equiv$	785204101	8366343965
$\Delta(x_2)\equiv$	9404708928	12046126152
$\Delta(x_3)\equiv$	16377832400	15562587599
$\Delta(x_4)\equiv$	23223121473	4774379183
$\Delta(x_5)\equiv$	3854722821	29205928729
$\Delta(x_6)\equiv$	24917588930	14195105338
$\Delta(x_7)\equiv$	12762587777	15670532757
$\Delta(x_8)\equiv$	28652706171	34094789793
$\Delta(x_9)\equiv$	18027024938	6297097808
$\Delta(x_{10})\equiv$	31321618182	6889674223
$\Delta(x_{11})\equiv$	18720580564	23699463484

これらの値から Fourier 変換によって, x^m の係数を求めると

$p=2^{35}-31$	$p=2^{35}-141$	
23342470269	28021907449	x^0
29629371739	29926920529	x^1
12116002332	12117870352	x^2
352302825	352331645	x^3
22349280914	22349280804	x^4
34010710490	34010710380	x^5
34359393753	34359393643	x^6
1435	1435	x^7
154	154	x^8
1	1	x^9
0	0	x^{10}

これらから答として

$$\Delta(x)=x^9+154\,x^8+1435\,x^7-344584\,x^6$$
$$-349027847\,x^5-12010457423\,x^4+9002603747119\,x^3$$
$$+583509192441266\,x^2+92942743873835032\,x$$
$$+1461674905790008175$$

例 2. 行列反転

$$(a)=\begin{pmatrix} 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 27 \\ 1 & 4 & 16 & 64 \\ 1 & 5 & 25 & 125 \end{pmatrix}$$

を反転すること.

これを mod$(2^{35}-31)$ で反転する経過を8進法表示で次に示す.

原方程式

$$\begin{aligned} 1\,x_1+\quad & 2\,x_2+\quad 4\,x_3+\quad 10\,x_4=y_1 \\ 1\,x_1+\quad & 3\,x_2+\quad 11\,x_3+\quad 33\,x_4=y_2 \\ 1\,x_1+\quad & 4\,x_2+\quad 20\,x_3+\quad 100\,x_4=y_3 \\ 1\,x_1+\quad & 5\,x_2+\quad 31\,x_3+\quad 175\,x_4=y_4 \end{aligned}$$

第 1 段

$$\begin{aligned} x_1+\quad & 2\,x_2+\quad 4\,x_3+\quad 10\,x_4=\quad 1\,y_1 \\ & 1\,x_2+\quad 5\,x_3+\quad 23\,x_4=377777\ 777740\,y_1+y_2 \\ & 2\,x_2+\quad 14\,x_3+\quad 70\,x_4=377777\ 777740\,y_1+y_3 \\ & 3\,x_2+\quad 25\,x_3+\quad 165\,x_4=377777\ 777740\,y_1+y_4 \end{aligned}$$

第 2 段

$$\begin{aligned} x_1+\quad & 5\,x_3+\quad 23\,x_4=377777\ 777740\,y_1+\quad 1\,y_2 \\ x_2+377777\ 777733\,x_3+377777\ 777703\,x_4=\quad & 3\,y_1+377777\ 777737\,y_2 \end{aligned}$$

$$
\left\{\begin{array}{rrrr}
 & 2x_3+ & 22x_4= & 1y_1+377777\ 777737y_2+y_3 \\
 & 6x_3+ & 74x_4= & 2y_1+377777\ 777736y_2+y_4
\end{array}\right.
$$

第3段

$$
\left\{\begin{array}{rrrr}
x_1+ & 11x_4=177777\ 777761y_1+377777\ 777740y_2+177777\ 777761y_3 \\
x_2+ & 30x_4= & 6y_1+377777\ 777731y_2+ & 3y_3 \\
x_3+377777\ 777707 & x_4=177777\ 777755y_1+ & 6y_2+177777\ 777756y_3 \\
 & 6x_4=377777\ 777740y_1+ & 3y_2+377777\ 777736y_3+y_4
\end{array}\right.
$$

第4段（結果）

$$
\left\{\begin{array}{rrrr}
x_1=052525\ 252520y_1+177777\ 777761y_2+177777\ 777760y_3+325252\ 525221y_4 \\
x_2= & 12y_1+377777\ 777715y_2+ & 17y_3+377777\ 777735y_4 \\
x_3=325252\ 525211y_1+ & 23y_2+177777\ 777741y_3+252525\ 252505y_4 \\
x_4= & 2y_1+177777\ 777753y_2+ & 5y_3+177777\ 777757y_4
\end{array}\right.
$$

$$
J\equiv 1.1.2.6.\equiv 14^2
$$

$$
A_{ik}\equiv J\cdot(a^{-1})_{ik}\equiv\begin{pmatrix}
170 & 377777\ 777361 & 264 & 377777\ 777661 \\
377777\ 777603 & 344 & 377777\ 777447 & 64 \\
30 & 377777\ 777637 & 74 & 377777\ 777717 \\
377777\ 777737 & 6 & 377777\ 777733 & 2
\end{pmatrix}
$$

（以上8進法）

これを10進になおし，正負の符号も正しくつけると

$$
(a^{-1})=-\frac{1}{12}\begin{pmatrix}
120 & -240 & 180 & -48 \\
-94 & 228 & -186 & 52 \\
24 & -66 & 60 & -18 \\
-2 & 6 & -6 & 2
\end{pmatrix}
$$

Vol. 1 No. 2 情 報 処 理 Sept. 1960

ディジタル計算機によるある制御系の準最適自動設計*

泥堂 多積** 伊藤 正美** 乗松 立木**

1. 緒 言

一般に複雑な系においては，その特性に影響を与えるパラメータの数が多く，しかも各パラメータが互にからみ合って系の特性を左右していることが多い．このような系を設計するに当って，最適なパラメータの組合せを求めることはかなり困難なことである．したがって設計は多くの場合，特性が仕様内におさまるようなパラメータの組合せを試行錯誤法的に見出すのが限度で，よりよい特性を与えるパラメータの値をさらに追求することは，時間的にも経済的にも許されない場合が少なくなかった．

近年，高速度・大容量のディジタル計算機の出現によって，複雑な系の最適設計も可能となって，これに関する文献もかなり発表されている．

筆者等の研究室で現在ある種の自動制御系について研究中であり，これに用いる補償回路の設計にディジタル計算機を使用し，系がある程度以上よい特性を示す補償回路のパラメータの組合せを自動的に求めた．その設計法と実施経験を主としてここに報告する．設計についての考え方，得られた結果の詳細と検討などについては他誌に報告する[1),2)]．なお，この設計法は単に Machine system の設計のみならず，Man-machine system の設計あるいは Decision making などにも適用できるものと考えられる．

* Design Synthesis Optimization of A Feedback Control System Using Digital Computer, by T. Deido, M. Ito and T. Norimatsu (Electrotechnical Laboratory)

** 電気試験所