

4. 線型計算 (連立一次方程式、L.P.)

鈴木 誠 道 (鉄 研)

§ 緒 言

電子計算機で行われる数値計算の中で、連立一次方程式、逆行列、行列の固有値問題、線型計画法 (L.P.) などの計算のしめる位置は大きい。電子計算機の発達とともに、この分野における多くの研究がなされ、新しい解法やアイデアが生み出されているが、なお多くの問題が残されている。

それらの問題は、もし、大容量、超高速、超精度の計算機があれば、ほとんど解決されるであろう。ところが我々は、これらのすべての面について、かなりの制約を受けている。従つてこの制約のもとに、その時々によつて、適当な方法を選び、各計算機の特徴を生かしたプログラムを作らなければならないが、又その経験を通して線型計算を行うためには、どんな計算機が望ましいかを計算機のデザインに反映させる事も必要である。

我々の所では、Bendix G 15-D を用いている。過去約2年の間に、線型計算のプログラムもある程度整備され、実用に供し得る段階に達している。プログラム・ライブラリーにあるプログラムは次の通りである。

1. 連立一次方程式

	方 法	プログラム方式	大 き さ	プログラマー
1. 1	GauB Seidel 法	Intercom	$n \leq 27$	鈴木
1. 2	消 去 法	Intercom	$n \leq 27$	鈴木
1. 3	消 去 法	Machine-Language	$n \leq 41$	鈴木
1. 4	Conjugate Gradient (C.G.法) 対称行列			鈴木
		Intercom	$n \leq 60$	鈴木

(B4)

対角項の左右の 5 要素以内にのみ non-zero の要素のあるもの。

1.5 C.G.法対称行列 Machine-Language

$n \leq 107$ 鈴木

Floating Vector を用いる。行列の要素は磁気テープに binary で store する。

1.6 消去法 Machine Language $n \leq 15$ Bendix

Double Precision の Floating Point の演算を用いる。

1.7 消去法, 逆行列 Machine Language $n \leq 32$ 鈴木

2. 固有値

2.1 Jacobi法, 対称行列 Intercom $n \leq 20$ 中西

2.2 Jacobi法, 対称行列 Machine Language $n \leq 32$ 中西

2.3 Power Method Machine Language $n \leq 107$ 鈴木

Floating Vector を用いる。行列の要素は、磁気テープに Store される。

2.4 Power Method Machine Language $n \leq 26$ 鈴木

高速演算装置を用いる。

2.5 Frame 法 Intercom $n \leq 10$ 鈴木

3. L. P.

3.1 輸送型 Machine Language $n+m \leq 56$ 鈴木

3.2 一般の L.P. Machine Language 条件式 ≤ 25 鈴木

改訂 シンプレックス法

変数 $< \infty$

Double Precision の Floating Point の演算を用いる。
初めのシンプレックス表は磁気テープに Store する。

3.3 一般の L.P. 条件式 ≤ 10

Intercom
シンプレックス法

変数 ≤ 40

丸山

以下において、連立一次方程式と L.P. について述べる。

§ 連立一次方程式・逆行列

Bendixが設置されて、最初につつたプログラムは、GauB-Seidelの反覆法とGauBの消去法であつた。プログラムになれるために、Intercomというinterpretive routineを用いた。しかしこれらのプログラムは、速度がおそく、消去法のプログラムがいくらか使用されたのみである。

その後、車両の構造計算から生じた30~60元の問題が持ち込まれた。これは対称行列で、対角項の左右に最大5要素non-zeroがある次のような行列であつた。

$$\begin{pmatrix}
 \Delta & \times & \times & \times & \times & \times & & & \\
 & \times & \Delta & \times & \times & \times & \times & \times & \\
 & & \times & \times & \Delta & \times & \times & \times & \times & 0 \\
 & & & \times & \times & \times & \Delta & \times & \times & \times & \times \\
 & & & & \times & \times & \times & \times & \Delta & \times & \times & \times & \times \\
 & & & & & \times & \times & \times & \times & \times & \Delta & \times & \times & \times & \times \\
 & & & & & & \times & \times & \times & \times & \times & \Delta & \times & \times & \times & \times \\
 & & & & & & & \times & \times & \times & \times & \times & \Delta & \times & \times & \times & \times \\
 & & & & & & & & 0 & & & & & \Delta & & & &
 \end{pmatrix}$$

Δ ; 対角項
 $\times$; non-zero の要素が来る位置

その当時は未だ磁気テープ装置はなく、プログラムや数値は、実際上すべて2000語程度のinternal storageにstoreしなければならなかつた。従つて、行列の特徴を取り入れ、その特徴が計算の途中で失われない方法、すなわち、くり返し法によつて計算する事にした。用いた方法は、C.G.法¹⁾であつた。

計算のAlgorithmは、周知の如く

$$AX=K$$

↓

$$P_0 = r_0 = K - AX_0$$

$$a_i = \frac{|r_i|^2}{(P_i, AP_i)}$$

(B4)

$$X_{i+1} = X_i + a_i P_i$$

$$r_{i+1} = r_i - a_i A P_i$$

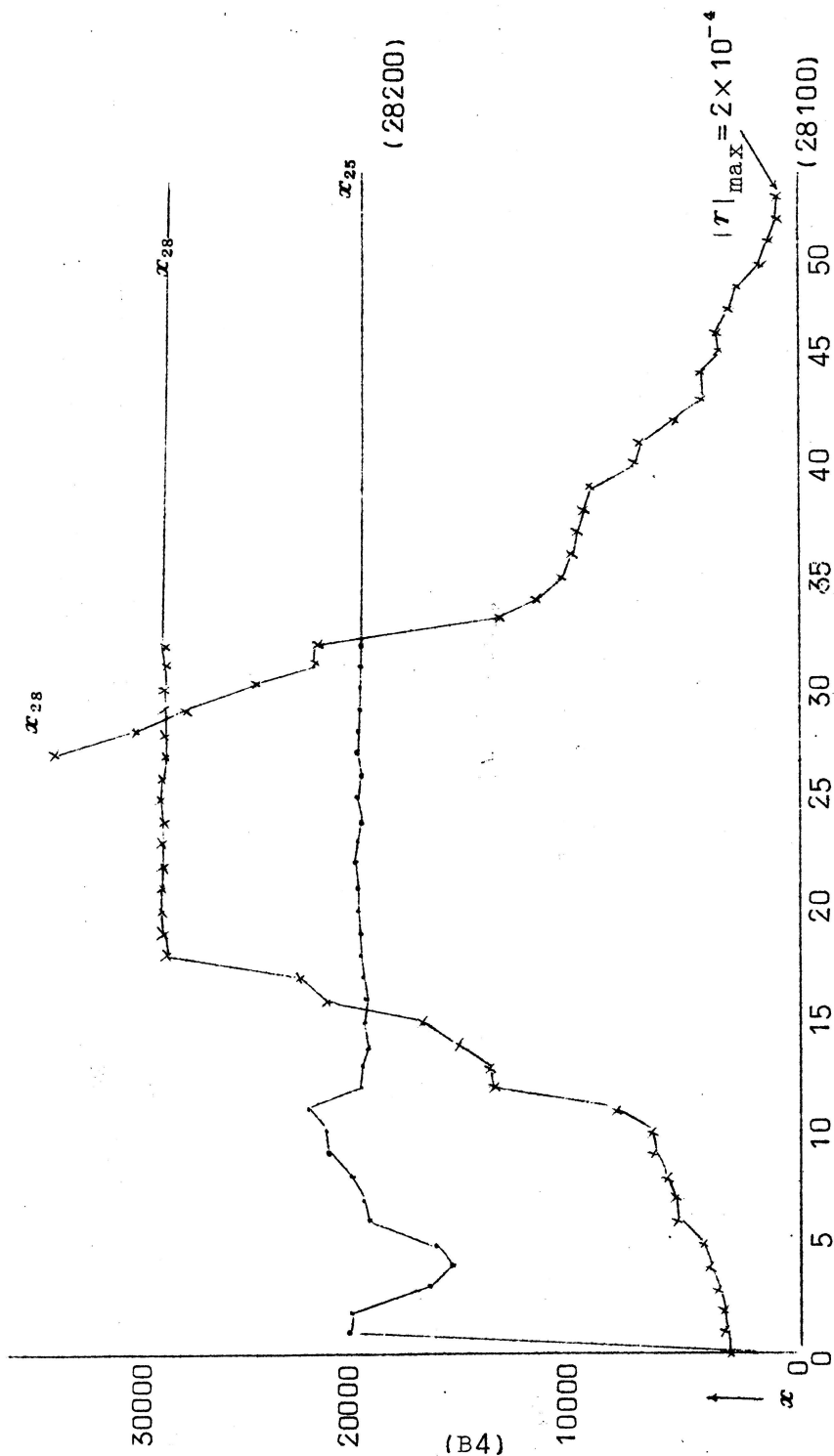
$$b_i = \frac{|r_{i+1}|}{|r_i|^2}$$

$$P_{i+1} = r_{i+1} + b_i P_i$$

である。

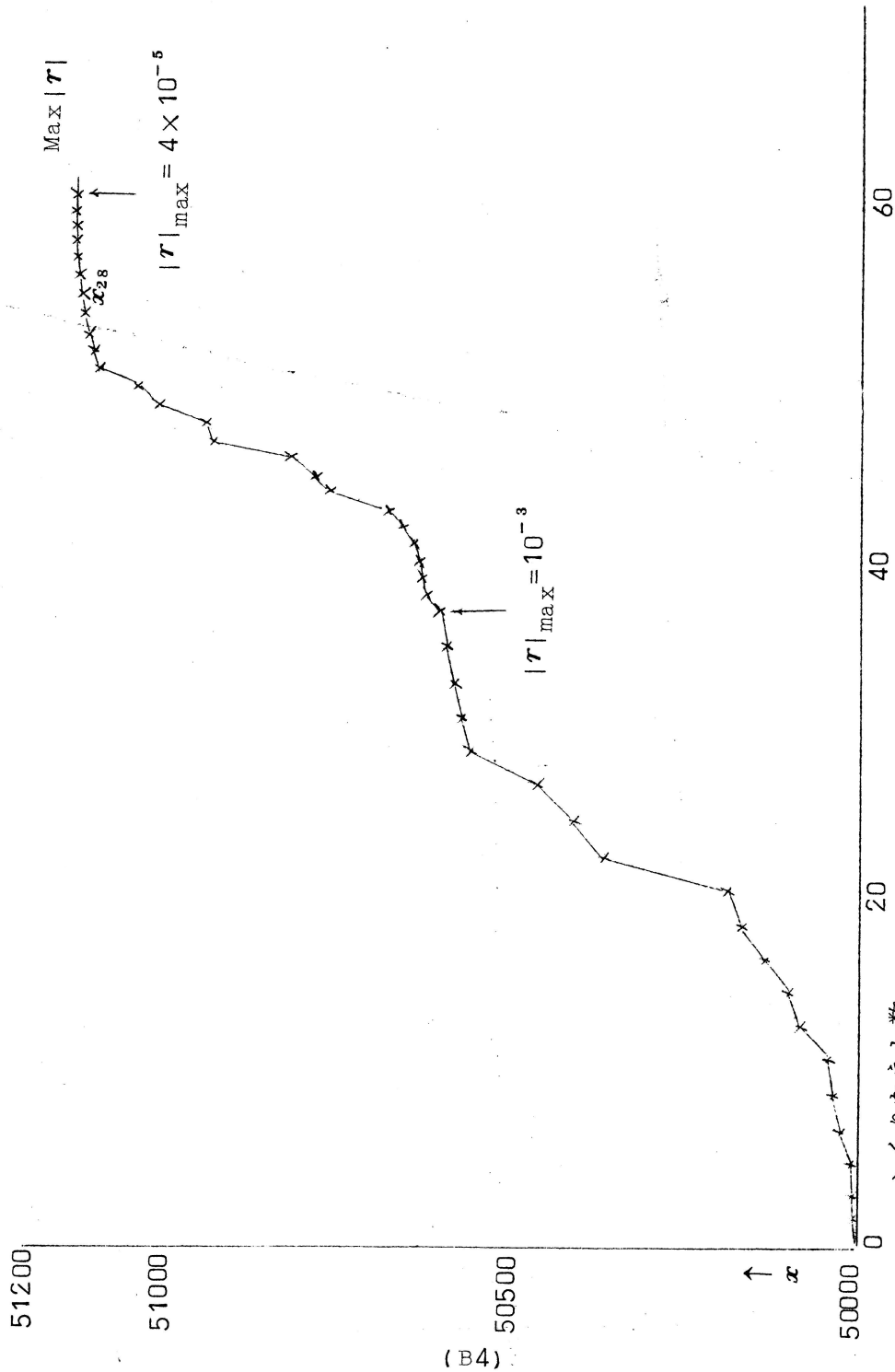
行列が特殊型である事、従つて計算時間が n^2 に比例するなどの事から Intercom (数は Floating Point で, mantissa に 21 bits, exponent に 8 bit が割り当てられている。) に対するプログラムを作り, 30 元の問題を数例計算して見た。

その中の 2 例について, ある変数についての解の収束のありさまを第 1 図 a, b に示す。



第1図a C.G.法の収束の様子(任意な値から出発した場合)

→くりえし数



(B4)

→ くりかえし数 第1図 b c.g. 法の収束の様子 (近似値から出発した場合)

a , b とも以前手計算で解の近似値が求まっていたのであるが, a では, X_0 を input する時4桁間違えてその $1/10$ を input した時のものである。27回以後のくりかえしにおけるありさまを右に目盛った scale に拡大して示してある。 b は, 近似値から出発したものである。2例とも, 1 iteration に要した時間は, 6~7分であつた。

この例や他のいくつかの例から結論出来るような事は,

1. 任意な X_0 から出発した場合は, 一般には, n 回のくりかえしでは, あまり高い精度の解は得られない, 通常有効数字2~3桁で, r の最大要素の相対誤差は $10^{-2} \sim 10^{-3}$ 程度である。
2. 1, 2桁の精度の近似解から出発した時は, 解の収束はおそく, n 回程度のくりかえしの後には, 全く任意な X_0 から出発したものに対しての有利さがほとんど失われてしまう。ただし, 容易に予想されるように近似値として, Conjugate Gradient 法で, あるくりかえしまでの結果として得られたものを input する場合は, このような事は起らない。
3. 有効数字4~5桁程度まで精度を上げようとするれば, ($|r|_{\max} \sim 10^{-4} \sim 10^{-5}$) 更に n 回程度のくりかえしが必要である。

この結果は, round off error によるものと思われる。之は20~30元のごく普通の行列に対するもので, すべての場合にこのようになるわけではない。5次程度の小さい行列や, 次数が大きくても(60元)たちの良い行列では, n 回かそれより数iteration 後には, 全く収束するか, 最小桁附近で異なるいくつかの解の間での周期的なくりかえしに達する。

行列の特徴を取り入れたこのプログラムでも4~5桁程度の精度で解を求めるのに, 30次の行列に対して, 5時間にも達す時間を要した。之では, 到底数多くの問題をこなすには, 不充分であつた。又もつと一般の行列に対するプログラムも必要であると思い, 消去法によるプログラムを作る事にした。

これは, Machine Language で, Fixed Point の演算を採る事に

して速度の向上を図った。方法は次の通りである。

$$\begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1^{(1)} \\ b_2^{(1)} \\ \vdots \\ b_n^{(1)} \end{pmatrix}$$

から出発し、

$$a_{ik}^{(r+1)} = a_{ik}^{(r)} - \frac{a_{ir}^{(r)}}{a_{rr}^{(r)}} a_{rk}^{(r)}, \quad b_i^{(r+1)} = b_i^{(r)} - \frac{a_{ir}^{(r)}}{a_{rr}^{(r)}} b_r^{(r)},$$

$$(r < i, k \leq n)$$

によつて

$$\begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ & & \ddots & \vdots \\ & & & a_{nn}^{(n)} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_n^{(n)} \end{pmatrix}$$

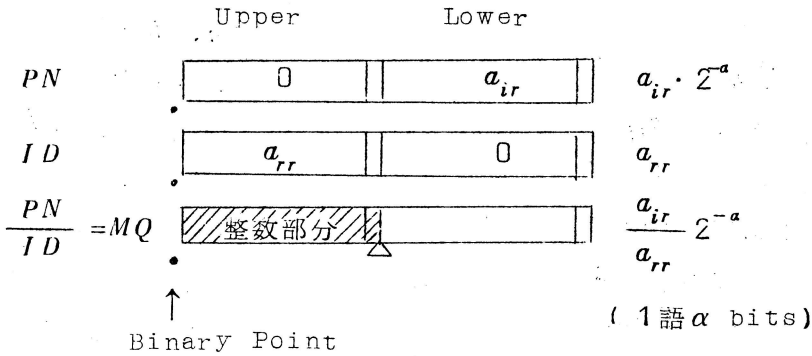
と変形した後、back solution

$$x_i = [b_i^{(i)} - \{a_{in}^{(i)} x_n + a_{i,n-1}^{(i)} x_{n-1} + \cdots + a_{i,i+1}^{(i)} x_{i+1}\}] / a_{ii}^{(i)}$$

によつて、 $x_n, x_{n-1}, \dots, x_1$ の順に解を求めるものである。

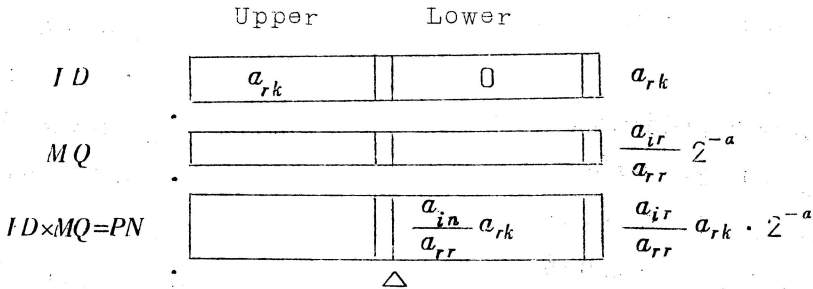
ここでよく問題になる点は、 $\left| \frac{a_{ir}}{a_{rr}} \right|$ が 1 を越える事、back

solutionにおける overflow の問題である。我々は、この困難を除くために、これらの部分にdouble precisionの演算を用いる事にした。すなわち、 a_{ir}/a_{rr} を求める時に、 a_{ir}, a_{rr} を乗除用の 2 語のレジスターに次のように入れて計算する。



$a_{rr} > a_{ir} \cdot 2^{-a}$ だから $\frac{a_{ir}}{a_{rr}} \cdot 2^{-a}$ は, overflow せず, $\frac{a_{ir}}{a_{rr}}$ の
 整数部分は MQ の Upper に現われる。

又 $\frac{a_{ir}}{a_{rr}} \times a_{rk}$ を求めるには, 次のようにする。



$\frac{a_{ir}}{a_{rr}} a_{rk} \cdot 2^{-a}$ は, $\cdot$ の処を binary point と考えた時である。従つ
 て PN の lower には, 丁度 $\frac{a_{ir}}{a_{rr}} a_{rk}$ が求まる事になる。なお之は
 double precision の乗算であるが, 時間は, single precision
 の時と同じで出来る。

back solution も上と同じ考えで処理する事が出来, x_i の整数部
 分が MQ の Upper に分数部分が lower に求まるから, まず Upper
 を整数として変換しタイプさせ, 次に lower を分数として変換させて
 タイプさせる。

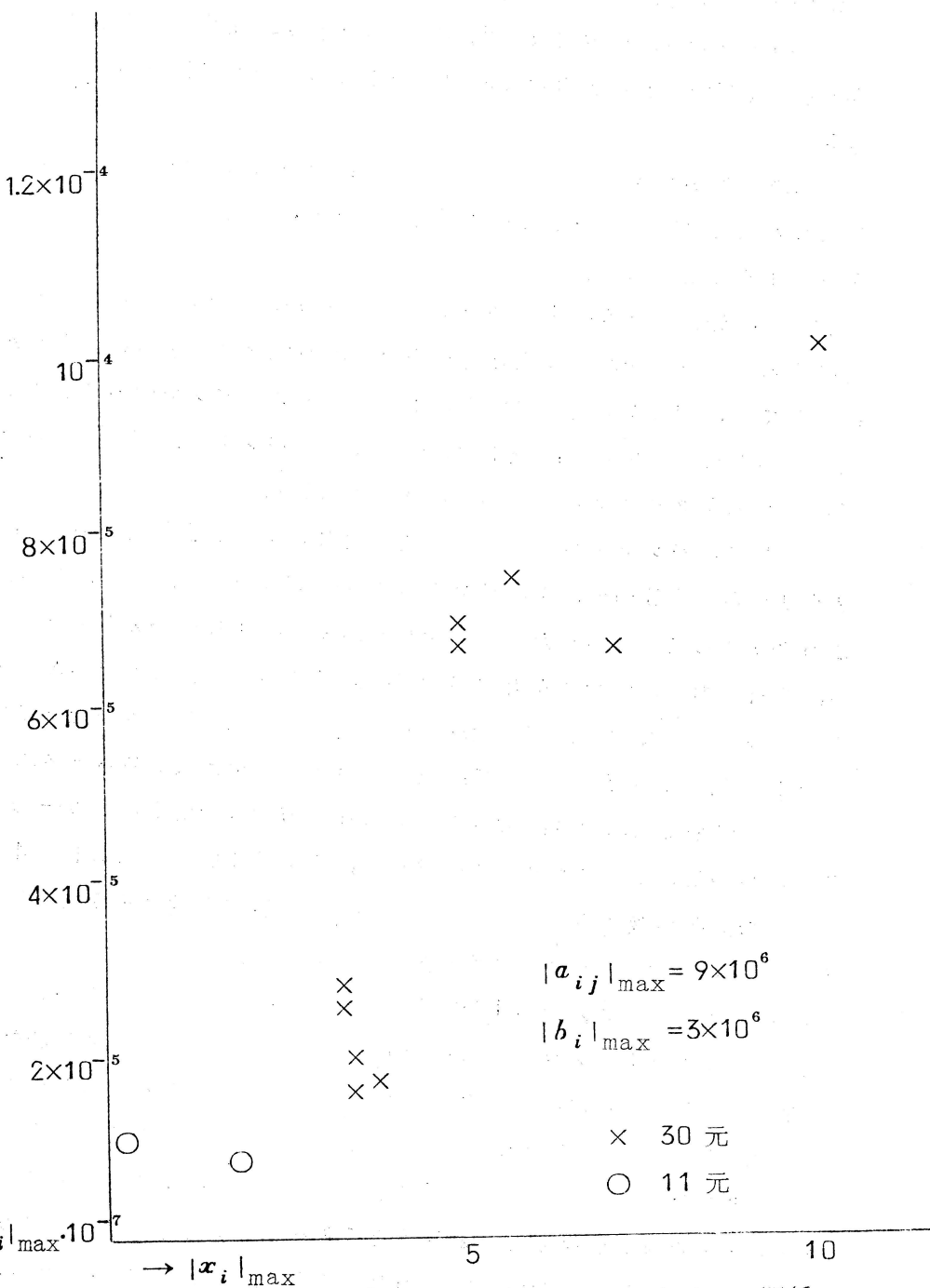
このやり方で問題になるのは, $a_{ik} - \frac{a_{ir}}{a_{rr}} a_{rk}$ の結果が single

precision に収まらなくなる事である。そのために、 $a_{ij}^{(1)}$ を store する時、most significant bits に少し余裕を作っておき、 a_{ij} が消去の操作で大きくなるのに備える事にした。すなわち、Bendix では通常 input を decimal の 7 桁で行うので、(1 語に収まるので 10 進→2 進変換に便利) $a_{ij}^{(1)}$ を 7 桁の整数として input する。そうすると、 $a_{ij}^{(1)}$ は $10^7 \cdot 2^{-28}$ を越えない。従つて most significant bits は 4.5 bits の余裕が出来る事になる。こうしても一般的には、 a_{ij} が容量を越える事があり得る。しかし我々は、ここで実際的には行列の要素が容量を越えるかどうかは、最初の行列の要素の具合によつて、比較的容易に判断し得るという見解を持つている。又今までの経験も之を保証している。このプログラムを用いて 30 元までの数多くの問題を解いた。

さて、消去法を用いて解を得た時、それが何桁目まで正しいかが問題になる。

C.G. 法などでは、収束のありさまから、之がある程度判断出来る。消去法の場合判断の手がかりになるのは余りである。それでこのプログラムでは X を求めた後、 $\tau = AX - b$ で τ が求められるようになっていゝる。我々の例では、 $|\tau_i|_{\max} \cdot 10^{-7}$ が $|x_i|_{\max}$ の相対誤差と余り異ならないという結果が得られた。又 $|x_i|_{\max}$ と $|\tau_i|_{\max} \cdot 10^{-7}$ の関係を示す第 2 図から $|\tau_i|_{\max} \cdot 10^{-7}$ が $|x_i|_{\max}$ にほぼ比例する事が云えそうだから、結局 $|x_i|_{\max}$ が大きくなると解の相対誤差がそれにほぼ比例して大きくなるという事になる。しかしこの点に関しては解の安定性 (A や b の値が変化した時、それが解にどの位の変化を与えるかなど) と関連して、なお理論的、実験的に調べて見なければ、はつきりした結論は出せないと思う。

ともあれ、このプログラムによつて Intercom に対する C.G. 法のプログラムで、30 元の問題に対して、4~5 時間程度要して得られる精度の解が、35 分程度で得られるようになり、この程度の問題を数多くこなす事が出来る条件の 1 つが満たされた。



第2図 $|x_i|_{\max}$ と $|r_i|_{\max}$ の関係

もう一つの条件は input であつた。之は連立方程式に限らず, input data の多い計算につきまとう問題であると思う。問題は数多くの data が能率よく準備され, 誤りなく計算機の中に store されるかどうかである。

行列の次数が大きくなると data の数は, ほう大となつて来る。しかし又 0 の要素の割合も多くなるのが普通であり, 何等かの形でこの事情を考慮する必要がある。

我々は, 磁気ドラムの 1 行分を data の一つの block と考え, その行に入る non-zero の要素とそれらの要素の位置を示す情報 (1 要素に対して 1 bit が対応している。) を紙テープにパンチすることにした。又対称行列の場合は, 上半分だけの要素をパンチし, input させて, それらをプログラムで下半分に移す事にした。

かくして, パンチすべき data の数が大巾に減り (30 元の例では, 930 から 150 程度に), パンチの手間やミスの発見が容易になつた。このようにして準備された data を読み込ませ, 10 進→2 進変換を行い, 1 行分の data の和を計算させて, タイプさせる。この和が人があらかじめ計算しておいた和と一致すれば, データが正しく入つたと見なすのである。この方式によつて, 数多くの data を比較的スムーズに

input 出来る事になり, 大きな次数の連立方程式を解く, もう一つの条件が整つた。このプログラムによつて, たちのよい行列に対して 41 元までの問題が扱えるようになった。 $a_{ij}^{(r)}$ が容量を越えそうな問題は, Floating Point のプログラムで行う事にした。

その後磁気テープ装置が 1 台設置される事が決定されたので, もつと大きな元数に対するプログラムを計画した。之には, くりかえし法を用いる事にした。それは, 大きな元数のものを消去法で行わそうと思えば, 計算機からの output が後に人手の介入なしに, しかも能率的に input 出来るような input output が欠くべからざる事, (このためには, 磁気テープ装置が少なくとも 2 台必要であろう。) 行列に多くの 0 の要素がある特徴を生かせない事などのためである。これに反し, 一般にく

りかえし法では、各 iteration 毎に初めの行列 A を用いるので、行列 A の性質が保たれる。又 iteration 毎に変化するのは、高々数ケのベクトルで之等は、次数が高くなつても、充分計算機の中に store 出来る。行列 A は、磁気テープに store しておき iteration 毎に、之を input すればよく、之には磁気テープ装置 1 台で充分である。

さて、くりかえし法では、行列とベクトルのかけ算、ベクトルの内積、ベクトルのスカラー倍、ベクトルの加算などの演算が計算の主要な部分を占めている。それで、之等の演算を floating vector²⁾を用いて、Subroutine に組む事にした。そうすれば、演算速度をそう落さずに、どんな場合にも overflow の心配のないプログラムが出来るからである。又行列 A を non-zero の要素とその位置を示す情報とで Compact に表わし、 Ax の計算で A の 0 の要素に対する演算を skip させる事によつて、 A の特徴を生かした。この subroutine を用いて、C.G. 法で 107 元まで扱えるプログラムを組んだ。之に対するいくつかの計算結果を挙げる。40~50% 0 の要素を含む 20 次の行列に対して、1 iteration 1 分程度、消去法で得られたものと同じ精度まで求めるのに約 40 回のくりかえしを要し、消去法の約 4 倍の時間がかかった。有効数字 7 桁まで正しく求めるのには実に 70 回のくりかえしを要した。

又 60 次の triple diagonal の行列

$$\begin{pmatrix} 5 & 4 & 0 & & \\ & 4 & 5 & 4 & 0 \\ & 0 & 4 & 5 & 4 \\ & & & \ddots & \\ & 0 & & 4 & 5 & 4 \\ & & & & 4 & 5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ \vdots \\ x_{60} \end{pmatrix} = \begin{pmatrix} 9 \\ 13 \\ 13 \\ \vdots \\ 13 \\ 9 \end{pmatrix}$$

に対して 1 iteration 2 分程度、30 回のくりかえしで 7 桁の精度の解が得られた。このような簡単な行列に対しては、ほとんど round off の影響がないものと思われる。解の収束も、 $|r_i|_{\max}$ が 28 回で 2^{-4} 、29 回で 2^{-7} 30 回で 2^{-12} 、31 回で 2^{-18} とこのあたりに来て急速に

収束して、29回では解の相対誤差が 2×10^{-2} であつたが30回では、 10^{-5} 、31回では 10^{-6} となつた。同じ行列で10次の場合には、この傾向はさらに著しく、4回では $|r_i|_{\max} \sim 2^1$ 、解の相対誤差0.3のものが、5回では $|r_i|_{\max} \sim 2^{-20}$ 、解は7桁目まで正しく求まつてしまつた。

以上が消去法、C.G.法に対する我々の経験の概要である。これらの経験から

我々は、消去法が予想以上に有効な事を認識した。C.G.法では、特殊な場合を除いてround off error がかなりの影響を与え、 n 回のくりかえしでは、2、3桁の精度しか得られない。消去法では $n^2/3$ 程度のelementary operation で一般にC.G.法以上の精度が得られる。ただ行列の次数が大きくなりかつ A の non-zero 要素の割合が減つて来ると、くりかえし法の有利さが發揮されて来るように思う。

又くりかえしが次数によらない Lanczosの方法³⁾も注目すべきものの一つであらう。

逆行列は、消去法で n ヶの連立方程式を同時に解く方法で解を求めている。

§ Linear Programming (L.P.)

L.P. は、一般のL.P. と特殊型のL.P. に大別される。特殊型の代表的なものは、輸送問題である。我々の所では、輸送型と一般型に対するプログラムを作つた。

(i) 輸送問題

解くべき問題は

$$\begin{pmatrix}
 a & \begin{pmatrix} 1 & 1 & \dots & 1 & 0 & 0 & \dots & 0 \\ & & & 1 & 1 & \dots & 1 & \dots \end{pmatrix} \\
 b & \begin{pmatrix} 0 & & & 0 & & & 0 \\ & 1 & & 1 & & & 1 \\ & 1 & 0 & & 1 & 0 & & 1 \\ & 0 & & 0 & 0 & & 0 & 1 \end{pmatrix}
 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{ab} \end{pmatrix} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_a \\ a_{a+1} \\ \vdots \\ a_{a+b} \end{pmatrix}$$

の条件のもとに

$$Z = c_1 x_1 + c_2 x_2 + \dots + c_{ab} x_{ab}$$

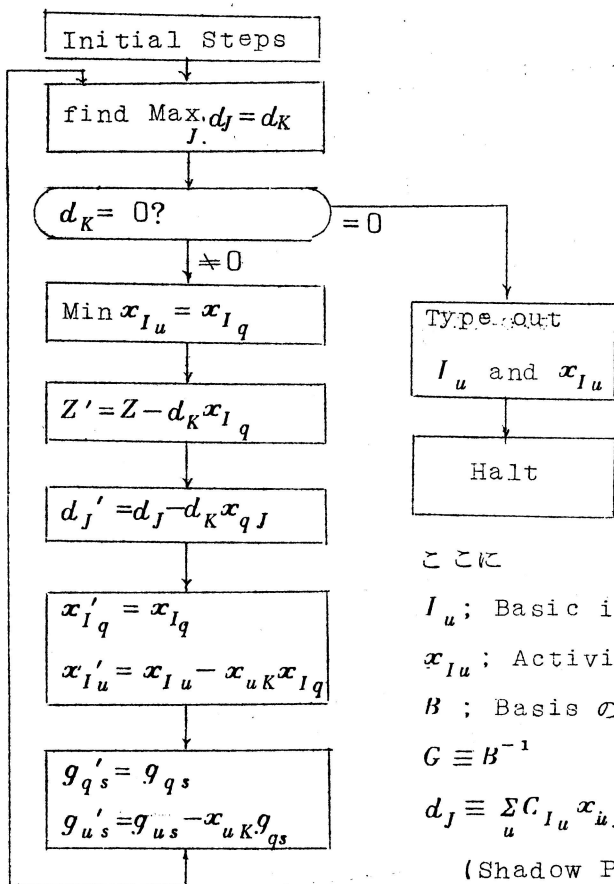
を最小にする事である。

解法は、改訂シンプレックス法によつて、basisの逆行列をiteration毎に変換して行く方法を探つた。そのalgorithmをFlow Diagramに示す。

このプログラムに用いているテクニックの原型は、D.G.Prinzの論文⁴⁾に見られる。

行列Gの要素は、0, 1, -1の三つの値しかとらない事を利用して、Gをcompactにstoreし、計算の途中に必要な x_{uk} を $x_{uk} = g_{ur} + g_{uv}$ なる演算によつて求めようとするものであるが、ここでは x_{uk} も0, 1, -1のいずれかであるので、この演算は特殊なternaryの演算になる。

我々のプログラム⁵⁾でも、この考えを取り入れ、Gの1要素を2 bitsで表わし、ternaryの演算を更に能率化した。このプログラムで扱える大きさは、 $a(\text{Source}) + b(\text{destination}) \leq 56$ で計算時間は、initial step(西北隅の原理)に約 $1.6^6 ab \text{ sec}$, iterative stepに $0.4 \sim 0.5 ab \text{ sec/iteration}$ であつた。



ここに

I_u ; Basic indices

x_{I_u} ; Activity level

B ; Basis の行列

$G \equiv B^{-1}$

$d_J \equiv \sum_u C_{I_u} x_{uJ} - C_J$

(Shadow Price)

第3図 輸送型 L.P. 解法の Flow Diagram

(iii) 一般の L.P.⁶⁾

問題は

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix} \quad (b_i \geq 0)$$

の条件のもとに

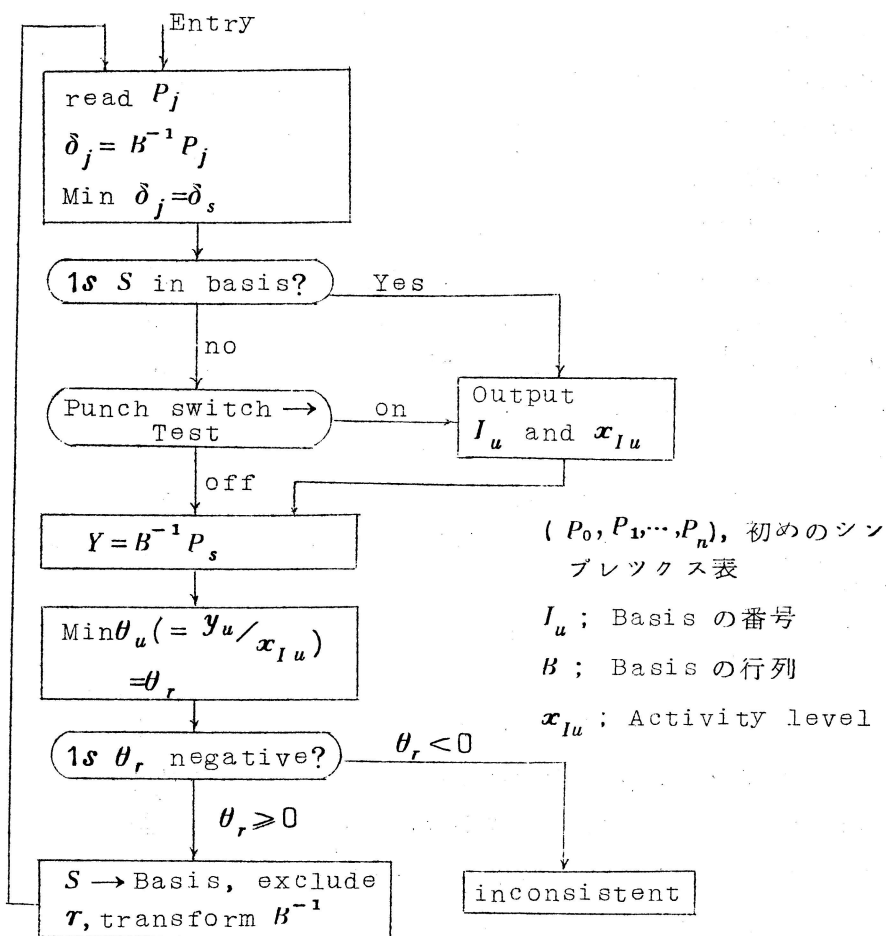
$$Z = a_{01} x_1 + a_{02} x_2 + \cdots + a_{0n} x_n$$

を最小にする事である。

プログラムを計画した当時は未だ磁気テープ装置が設置されていなかったもので、計算の途中で変化して行く量は、全て計算機の中に store しなければならなかつた。従つて、解法としては、Standardなシンプレックス法か、改訂シンプレックス法で、Basis の逆行列 B^{-1} を store しておく方式かのいずれかを選ぶ事になつた。ここで我々は、改訂シンプレックス法を採用した。それは、シンプレックス表を store するには、 $m n$ 単位、 B^{-1} を store するには m^2 単位の memory が必要であるが、一般には $n > m$ であるので、必要な memory の点で改訂シンプレックス法が有利な事、 B^{-1} を store する方式で要求される iteration 毎の初めのシンプレックス表の input を Ferranti の high speed tape reader で行える見透しがたつたからである。(この部分は、後に磁気テープからの input に置きかえた。)

改訂シンプレックス法の計算の Algorithm を Flow Diagram に示す。

演算はすべて double precision の Floating Point で行う事にした。最初は、 I_u , x_{I_u} , B^{-1} を計算機の中に store し、 $\{P_j\}$ は、あらかじめ binary で紙テープに取つておき、之を loop にして読み込ませるように計画した。之によつて条件式 25 までのものならば、どんな多くの変数のものでも解く事が出来るようにした。この方法で、問題になるのは、 $m n$ 回の乗算を含む $\{B^{-1} P_j\}$ の計算である。ここで注目すべきは、通常 $\{P_j\}$ には、多くの 0 が含まれているという事である。この特徴をうまく採り入れるか否かは、この計算が能率よく行われるか否かに直接関係する。そこで、我々は P_j をその non-zero の要素の数値と、それが何番目の要素であるかを表わす 1 要素当り 1 bit の情報とで Compact に表わす事にして、 P_j の読み込みの時間、 $B^{-1} P_j$ の演算時間を短縮しようとした。この面では、確かにこの意図は成功した。



第4図 一般型L.P. 解法の Flow Diagram

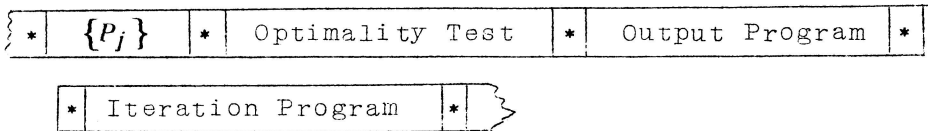
しかしもう一つの困難があつた。それは $\{P_j\}$ の読み込みを紙テープによつて行つたからである。 $\{P_j\}$ を Compact な形にしても、変数が多くなると紙テープの長さがぼう大(変数200の例で、50m位になつた。)になり、テープの読み込みがスムーズに行かなくなつた事、read back の機構がないので、読み誤つた時、これをプログラムで読み直させようとするとは非常に能率が悪くならざるを得なかつたからである。又紙テープだけを用いたのでは、計算部分のプログラムと output

のプログラムの相互の交換が要領よく行われず、計算の進行を見るのが容易でなかった。結局この方式では、人が注意している事が必要で、使いつらいプログラムであつたため、2, 3の例が計算されたに過ぎなかった。

その後、磁気テープ装置が設置されたので、上述の紙テープの部分を磁気テープに store しておき必要に応じて、プログラムの入れ換えが出来るようにした。この改良によつて $\{P_j\}$ の読み込み、計算進行の様子を見る事がスムーズに行われるようになり、人手がほとんどかからなくなった。

磁気テープの使い方は、次の通りである。

→ Forward



* File Code

まず P_{j+1} を読みながら $B^{-1}P_j = \delta_j$ を求め、 $\min \delta_j = \delta$ が求まつたら、Optimality Test のプログラムを読み込み、Optimalであれば、次のOutputのプログラムを読んで結果をタイプさせる。Optimalでなくとも、Punch Switch が on になつていれば、Outputのプログラムに入つて、計算進行の状態を見る事が出来る。Punch Switch が off ならば、File Code を2つさがして、read back させテープをもとの位置におく。一方Outputに入つた場合には、Outputの後又 Iteration の Program を読ませると同時に、File Code を4つさがしてread back させ計算を続行させる。磁気テープに書き込まれている情報は、各1ブロック毎の和が0になるようにBalance されていて、もし読み誤りがあれば、File Code の処までBack させて、(45 inch/sec) 読み直させる (7.5 inch/sec, 約400CH/sec) ようにしてある。このように磁気テープを使う事により、プログラムは、大分使い易いものになった。

そこで数多くの問題を計算した。条件式は20~25, 変数は, 60~200程度であつた。計算時間は, 20×100 のもので $B^{-1}P_j$ の計算に約1分, B^{-1} の変換に約2分, 結局1 iteration 約3分, outputをさせると1 iteration 約5分を要した。 25×200 で $B^{-1}P_j$ に約2.5分, B^{-1} の変換に約3.5分, 結局1 iteration 約6分 outputさせると約9分を要した。この例からも $B^{-1}P_j$ の計算時間がかかなり短縮されている事がわかる。精度は, 有効数字11~12桁である。ただし問題を次のような標準型に直した時

$$\begin{pmatrix} 1 & M & 0 & M & a_{0m+1} & \cdots & a_{0n} \\ & 1 & & & a_{1m+1} & & a_{1n} \\ & & 1 & & \vdots & & \vdots \\ & & & 1 & \vdots & & \vdots \\ & & & & \ddots & & \vdots \\ & & & & & 1 & a_{mm+1} \\ 0 & & & & & & a_{mn} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} 0 \\ b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

等式の条件に対応する Slacks には, Pentaly M を課す方法を採用した。この時余り M を大きくすると $Z(x_0)$ の精度が幾分落ちるようである ($M/a_{0j} \sim 10^5$ で, Z の精度が8~9桁) 条件式25以上に対するプログラムは未だ作っていないが, 之は Product Form for Inverse⁷⁾の方法によるのが一番よいと思う。検討した結果, 磁気テープ1台でも充分間に合うので現在計画中である(条件式100程度)。又この方法は, 大きな次数の行列の逆転や連立方程式の解を求める場合にも有効に使えるので, 今後一層注目されるように思う。

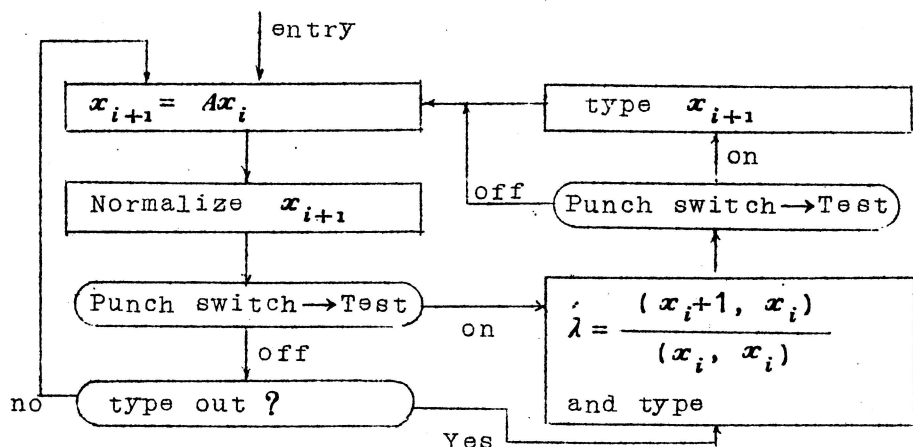
§ 演算高速化装置⁸⁾を用いての試み

演算高速化装置は穂坂氏が鉄研在任中に設計された Bendix G15-D に対する附加装置である。Bendixに從來あつた命令に加えて, 27種類の新しい命令を導入し, Floating Point の演算 (Single precision) や, 乗算の高速化を行わせると同時に block command の考

えを用いて、内積、最大値選択、ベクトルの和などを一つの命令で 1 drum revolution 内に行わんとするものである。

この 11 月末 (59 年) に設置され、12 月に入ってから順調な動作を開始したので、未だ多くの経験を得るに至っていない。簡単な例として Power method で固有値を求めるプログラムが完成したので、この装置が線型計算に対していかに有効かを示す一例として、之を説明するとともに用いられた命令について若干ふれようと思う。

Power method による計算の手順は次の通りである。



第 5 図 Power Method の Flow Diagram

この計算に用いた特殊な命令について説明する。

(i) 内 積

ドラムの 2 つの行 (06, 07 行各 108 語) の対応した場所にある数の積を 08 行の対応した場所に書き、積の和を 2 語のレジスター PN に出す。すなわち

09	08	07	06	05	04	03	02	01	00	
		a_4		a_3		a_2		a_1		06
		b_4		b_3		b_2		b_1		07
$a_4 \times b_4$	$a_3 \times b_3$	$a_2 \times b_2$	$a_1 \times b_1$							08
$\sum a_i b_i$										PN

である。この命令で最高53要素の内積が 1 drum revolution (29 m.s.) で出来る。

(iii) 最大値の選択

08行にある数の最大値が、1語のレジスター ER に、その location を表わす数が1語レジスター AR に残る。

(iii) Block Shift

08行にある数が同時に 1 bit 左へ又は右へ Shift される。

上の三つの特殊な命令の中、 Ax_i には内積を、 x_{i+1} の normalization には、最大値の選択と Block shift を、 λ には内積を用いた。 A の要素を計算機の中に store するように考え、最高26次まで扱えるようにした。1 iteration に要する時間は n^2 に比例せず、 n に比例する。26次の行列で 1 iteration 4 sec. 程度であつた。もし之を従来の命令のみで行わんとすれば、1要素当り 5 drum revolution として、2~3分程度を要するから、従来の計算速度に比して実に、30倍から50倍の速度になる。

この例は、この装置がフルに能力を発揮する計算の1例であるが、他の線型計算についても同程度の効果が期待出来るように思う。ただ計算機の容量を越える問題では計算の速度が入出力の速度でおさえられてしまうので、従来の10倍程度の速度が期待されるように思う。

§ 結 び

線型計算で、問題になるものの1つは、大きな次数の問題をいかに処

理するかであろう。

それには、高速で入出力の能力の高い計算機が望まれるのはもちろんであるが、その他に行列の特徴を生かせるような工夫をする事、線型計算に適合したLogicをhardwareとして附加する事などが必要であろうと思う。かくすれば、演算が能率化する事はもち論、本質的に重要でないアドレス変更などにわずらわせずに、計算をもつと大づかみに把握する事が出来、プログラムも楽になるように思う。

他の1つは、ぼう大なinput dataをいかに準備するかという事である。そのためには、行列の要素の数値を求める仕事を出来るだけ自動化する事が必要であろう。すなわち、要素の数値を人が一、一求めるのではなしに、その数値を決定させるより基礎的なデータから計算機の中で、それを計算させるような方法が望ましい、限られた問題に関しては、すでにこのような方法が採られているが、未だ充分でないように思う。この事は、計算機で行われる計算相互についても云える事で、ある計算の結果が他の計算のinputになる時、数の受け渡しがスムーズに行かなければならない。例えば、 A から A^{-1} を求め A^{-1} の固有値を求めるような場合である。

§ 附

以上述べた結論を他の計算機に対するものと比較する時の参考のために鉄研にあるBendix G45-Dの性能を記す。

名称; Bendix G45-D

数表示; 28 bits+1 sign bit or 57 bits+1 sign bit

型式; serial, binary

命令; 特殊な2アドレス方式

演算時間; 加減算 0.27m.s. 乗除算 16.7m.s.

内部記憶; 磁気ドラム

1 Word line 1行

2 Word line 5行

(B4)

4 Word line 4 行 (平均 access 0.54m.s.)
108 Word line 20 行 (平均 access 14.5m.s.)
入出力; タイプライター 8CH/sec.
フオートテーブリーダー 200CH/sec.
Ferranti のフオートテーブリーダー 430CH/sec.
磁気テープ装置 (1 台) 430CH/sec. (読み書き)
430×6CH/sec. (Search)
テープパンチ 18CH/sec.

参 考 文 献

- 1) M.R.Hestenes and E.Stiefel; Method of Conjugate Gradient for Solving Linear Systems.
J.Research N.B.S., Vol.49, No. 6, Dec. '52.
- 2) 鈴木; Floating Vector について, プログラマー懇談会資料,
Sept. '59.
- 3) C.Lanczos; Iterative Solution of Large-Scale Linear Systems, J. of Industrial and Applied Mathematics,
Vol.6, No. 1, '58.
- 4) D.G.Pring; Programming the Transportation Problem for Digital Computer, Technical Notes CS 65, Ferranti, Moston, Manchester, '55.
- 5) S.Suzuki; Programming Technique for Solving the Transportation Problem, J.Operations Research. Society of Japan, Vol.1, No. 4, June '58.
穂坂; ORおよび制御工学における応用, 電子計算機講習会テキスト, 物理学会, '59年8~9月
- 6) 鈴木; Bendix G15-D に対する L.P. のプログラム, O.R. 学会講演会, 59年11月
- 7) G.B.Dantzig and W.Ochard Hays; the Product Form for the Inverse in the Simplex Method, M.T.A.C., Vol. VIII, No. 46, April '54.
- 8) 穂坂; 未発表

本 PDF ファイルは 1960 年発行の「第 1 回プログラミング・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、権利者の検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なからずありました。

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、一定期間の権利者搜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>