UDC 512.8
518.5
681.142

航空宇宙技術研究所資料

TECHNICAL MEMORANDUM OF NATIONAL AEROSPACE LABORATORY

TM-121

大きなマトリクスの逆行列計算および連立一次方程式の計算のためのプログラミング技術

戸川隼人・戸川保子

1967年11月

航空宇宙技術研究所
NATIONAL AEROSPACE LABORATORY

既　刊　資　料

TM-76	コーティングの断熱効果のアナログシミュレイション	1966年2月	小川鉱一
TM-77	テレメータ電波の偏波面の回転を利用したロケットのスピン測定について	1966年3月	田畑浄治，桜井善雄 三浦雅男
TM-78	昇降舵の操舵力特性に関するシミュレータ解析	1966年3月	堀川勇壮，森　幹彦 中野佳直
TM-79	テレメータ機上装置の小型化の研究	1966年3月	新田慶治
TM-80	安定制御のための一計算法	1966年3月	楢崎哲二
TM-81	吹出式超音速風洞の起動時および停止時における過負荷防止装置	1966年3月	石原久蔵，斎藤秀夫 外立政隆，榊原盛三 宗美　均
TM-82	ピトー管による境界層速度分布測定について	1966年4月	長洲秀夫，柏原登喜子
TM-83	ジェットリフトエンジン空気取入口の実験（II）	1966年4月	近藤　博，増田惣平
TM-84	二段ロケットの低速風洞試験	1966年7月	毛利　浩，田村敦宏 佐野四郎，能村　実
TM-85	航空機の滑走路走行時の振動に関する実験的研究	1966年8月	小野幸一
TM-87	極小型超音速機用姿勢制御装置の特性解析	1966年8月	池谷光栄，畑山茂樹
TM-88	プロペラ後流偏向型 STOL 機の風洞試験（I）	1966年9月	犬丸矩夫，岡部祐二郎 北村清美，川幡長勝 木村友昭
TM-89	有孔板の振動について	1966年9月	川井忠彦，泉　日出夫
TM-90	地上付近の横風の影響による小型ロケットの方位角変化	1966年9月	戸川隼人，石黒登美子
TM-91	高速タービン翼列二次元試験	1966年10月	近藤　博，蓑田光弘 山崎紀雄
TM-92	リフトジェットエンジン試験設備（I）—台上運転設備—	1966年10月	大山耕一，吉田　晃 中山　晋，菅原　昇 五味光男
TM-94	J-3ジェットエンジン用タービン動翼の固有振動特性	1966年11月	武内澄夫，宮地敏雄 星谷昌二
TM-95	超軽量軸流圧縮機動翼の固有振動特性	1966年11月	武内澄夫，宮地敏雄 星谷昌二
TM-96	2024-T4および7075-T6有孔平板の曲げ疲労試験と2024-T4平滑丸棒の軸荷重疲労試験	1966年11月	佐野政明，孤岡一洋
TM-97	高マッハ数風洞について（II）	1967年1月	吉永　崇，井上建二 広田正行，楯　篤志
TM-98	40kWプラズマ発生装置の諸特性	1967年1月	野村茂昭，相原康彦
TM-99	搭載機器用環境試験装置の特性	1967年2月	大月正男，鈴木孝雄 田畑浄治，円居繁治
TM-100	二連型リフトエンジンの吸込み抵抗	1967年3月	近藤　博，大城章一郎
TM-101	低圧環境下における固体ロケットモータの性能	1967年3月	望月　昌，斎藤　信 五代富文，伊藤克彌 湯沢克宜
TM-102	弾性支持片持板の振動	1967年3月	塙　武敏，築地恒夫 越出慎一，林　洋一
TM-103	結合梁の振動について	1967年3月	築地恒夫，林　洋一
TM-104	平板翼模型の固有振動モードの測定	1967年4月	中井暎一，森田甫之
TM-105	非定常境界層方程式を含む放物型微積分方程式の数値解法	1967年4月	関口清子

目　　次

---

*　Conjugate Gradient method（共役勾配法）
**　Steepest Descent method　（最急降下法）
***　Gauß-Seidel's method　（ガウスザイデル法）

大きなマトリクスの逆行列計算および連立一次方程式の計算のためのプログロミング技術*

戸川隼人**・戸川保子**

構造解析の計算などに必要な，大きなマトリクスの逆行列計算または連立一次方程式の計算を，大型電子計算機により，効率よく計算するためのプログラミング技術に関し，調査，研究した成果をまとめたもので，あわせてその応用として当所の大型電子計算機 HITAC 5020のために開発した29種のサブルーチンについて使用法を記した。

ま え が き

この資料は，大きなマトリクスの逆行列計算，または大きな係数行列を持つ連立一次方程式を解く計算を，大型電子計算機で処理するためのプログラミング技術について論じ，また，その応用として，当所で使用している大型電子計算機 HITAC 5020 のために作成した29個のサブルーチンについて述べる。

ここで対象とする「大きなマトリクス」とは，次数 (order) n が50以上のマトリクスを意味するものとし，一応 $n=1,000$ ぐらいまでの問題を中心に考えることにする。その理由は，n が50以下の問題に関しては，HITAC 5020 の標準サブルーチンとして，連立一次方程式を解く LINSW (掃出法)，DLINSW (同，倍精度計算)，LINGS(ガウスザイデル法)，DLINGS (同，倍精度計算)，LINCG (共役勾配法)，DLINCG (同，倍精度計算) および逆行列を計算する INV (掃出法)，DINV (同，倍精度計算) などがあり[1]，これらを使用すれば，よほど性質の悪い問題でない限り，実用上十分な精度で，比較的短かい時間で (たとえば $n=50$ の時の逆行列計算でも45秒程度) 計算することができる。これに対して，$n>50$の問題に関しては標準サブルーチンがなく，自分でサブルーチンを作成するとしても，大きなマトリクスはコアメモリに収容できず，外部記憶装置を使用する必要があるので，プログラムが複雑になり，また，マトリクスが大きければ長大な計算時間を要するので，プログラムを工夫して少しでも計算時間を短縮することが必要になり，このために高度なプログラミング技術が要求されることになる。

航空関係で，この種の大規模な計算を必要とする代表的な分野は，機体の構造解析の計算である。特に，複雑な構造物の解析の有力な手段として注目されているマトリクス法を用いて，航空機の胴体，翼などの応力解析，たわみの計算，振動の計算などを行なう場合，数百次以上のマトリクスの逆行列計算や連立一次方程式の計算が必要になる。この種の問題に現れるマトリクスの特徴は，零要素 ($a_{ij}=0$ のもの) が多く，かつ，それがかなり不規則に並んでいる，という点にある。本資料に述べるプログラミング技術およびその応用サブルーチンは，主として上記の形の，構造解析関係のマトリクスに応用することを目標にしている。この結果を他の分野に適用することは，もちろん可能であるが，問題によってはほかにもっと適切な方法が存在するかも知れない。たとえば偏微分方程式を差分法で解く場合に現れるマトリクスは，やはり零要素が多いが，非零要素の配列に規則性があり，また非零要素の値自身に関しても，かなりの規則性があるので，その固有な性質を利用した優秀な解法がいろいろ研究されている。しかし本資料では構造解析以外の特殊な問題には立入らない。

本資料で述べる「プログラミング技術」は，特に斬新で画期的なものではなく，どちらかといえば平凡なアイディアの積み重ねである。しかし，この平凡な工夫をするかしないかでプログラムの効率に著しい差が生じる。たとえば著者らが扱った500元連立一次方程式のある例においては，普通の方法で315秒かかる問題を全く同じ数値計算公式を使用して，プログラムだけ

* 昭和42年10月24日受付
** 計測部

を改良した結果，わずか4秒で解くことができた。計算センタで見ていると，利用者の中には「普通の方法」程度のプログラムさえも書けていない場合があり，たとえば，磁気ドラムから1ワードずつデータを取り出すというような非常に能率の悪いプログラムを書いて「普通の方法」のさらに百倍ぐらいの時間を浪費している例がある。したがって，たとえ平凡なアイディアであっても，この種のプログラミング技術を資料としてまとめておくことは，計算機の能率的使用のためには非常に有意義なことであると考える。

また，これらのプログラミング技術を応用して作成した29個のサブルーチンは，単なる応用例としてのサンプル・プログラムではなく，十分に実用に耐えるものであり，$n=50$ないし1,000の範囲の各種の問題に適用することができる。収録したプログラムは目次に示したとおりで，全体として一つのセットとして使用できるように設計されており，単に1個の方程式を解くだけでなく，前後の計算とのつながりが円滑に処理できるよう配慮されている。

本資料は3部から成り，第1部は基礎となるプログラミング技術一般の説明，第2部はその応用サブルーチンの仕様説明，第3部は使用例と所要時間例である。

第1部 プログラミング技術

1.1 大きなマトリクスを扱う場合の問題点とその対策の概要

大きなマトリクスを扱う場合の主要な問題点は

時間の問題……計算時間が非常に長くかかる

精度の問題……十分な計算精度が得られない

容量の問題……計算機の記憶容量が不足する

などである。

計算時間は，計算方式を一定とすれば，マトリクスの次数nの三乗にほぼ比例する。HITAC 5020 の場合，掃出法倍精度計算で連立一次方程式を解くに要する時間は，$n=50$ の場合に約16秒であるから，$n=500$ の場合には（仮にコアメモリが無限にあり精度の問題も考えないとしても）約5時間，$n=5,000$の場合には約5,000時間かかることになる。これではなんとも時間がかかりすぎるので，計算時間をできる限り短縮する工夫が必要となる。

計算時間を短縮する方法としては，プログラムを工夫することと，適切な数値解法を用いることとの，二通りの手段が考えられる。プログラムを改良する主眼点は，

(1) 特殊な形のマトリクスは，その特性を利用してむだな計算を省略するようにプログラムを書く。

(2) FORTRAN で能率の良いプログラムを書けない部分は機械語（またはアセンブラ言語）を用いてコーディングする。

(3) 外部記憶装置の待時間等による時間の損失を少なくするようにプログラムを書く。

などである。一方，数値解法に関しては，

(1) 大きなマトリクスの場合には，掃出法よりも反復法の方が有利になることが多いので，これをよく検討する。

(2) 特殊な形のマトリクスに関しては，その特性を生かした解法を用いる。

(3) 外部記憶装置とのデータのやりとりの回数が少なくてすむような解法を用いる。

などが重要なポイントになる。

次に計算精度の問題。これはマトリクスが大きい場合に必らず困難になるという性質のものではないが，一般的に言うならばマトリクスが大きくなるほど悪条件の問題（たとえば計算中に有効数字の桁落ちなどが起る問題）に遭遇する確率が大きくなる。この対策としては，

(1) 有効数字の損失をできるだけ予防するような慎重なコーディングをする。

(2) 検算を厳重に行なう。

(3) 精度不十分の場合に，精度を改良できるようにそのためのサブルーチンを準備しておく。

などの，きめのこまかい配慮が必要である。

次に容量の問題。マトリクス全体がコアメモリに収容できれば問題ないのであるが，たとえば当所の大型計算機の場合（昭和42年8月現在の構成で），記憶装置の種類とそれに収容できるマトリクスのサイズの限界は，およそ下記のとおりであって，これによって各種の記憶装置を使い分けなければならない。

種類	待時間	サイズの上限*
磁気コア	2マイクロ秒	150(100)
磁気ドラム	0～10ミリ秒	360(250)
磁気テープ	5ミリ秒～3分	1,500(1,000)

このようにマトリクスの大きさによって装置を使い分けることは煩雑であるが，時間効率の点からやむを得ない。

しかし特殊な形のマトリクスに関しては，記憶容量を節約することができるので，上記の制限にかかわら

* （ ）内は倍長語の場合。磁気テープに関してはリール1本当り。

ず大きなマトリクスをコアメモリ内部だけで処理できて好都合である。たとえば

(1) 零の多いマトリクスは，零でない要素だけを記憶するようにする。

(2) 対称行列は半分だけ記憶する。

このようにすれば時間的，容量的に有利になるがプログラムは複雑になる。

以上述べた内容を「対策」の面から要約すると

(1) 特殊な形のマトリクスはその特性をなるべく利用する。

(2) 外部記憶装置を能率よく動かす。

(3) 機械語を用いたコーディングをして時間短縮を計る。

(4) 計算精度の保持，検査，改善について留意する。

(5) 数値解法の選択を適切に行なう。

などとなる。これらについて以下順を追って説明する。なお(1)の「特殊な形」としては，まえがきにも述べたとおり，零要素の個数が多くその配列が不規則な場合を中心にして考察する。

1.2 零要素を多く含んだマトリクスの計算を合理化する方法

〔ゼロジャンプ法〕 零要素の多いマトリクスの計算の時間を短縮する最も簡単な方法は，個々の計算を実行する前にデータが零であるか否かを検査し，もしも零ならばそれに関する $0\times a$ とか $0+a$ などの計算を省略するという方法である。仮にこれをゼロジャンプ法と呼ぶ。

(例1) N×NマトリクスAをベクトルXに掛けて結果をYとする計算。

```
      DO  1  I=1, N
      Y(I)=0
      DO  1  J=1, N
      IF (A(I,J).EQ.0.) GO TO 1
      Y(I)=Y(I)+A(I, J)*X(I)
1     CONTINUE
```

4行目のIFが $a_{ij}=0$ を判定するステートメントで，これがなければ普通のプログラムである。このループ1回当りの所要時間は

$a_{ij}=0$ の場合　　44μs

$a_{ij}\neq 0$ の場合　　150μs

である。いまAの全要素のうち，$a_{ij}=0$ のものの個数をM個として，$\alpha=M/N^2$ とすれば（すなわち零要素の割合を α とすれば），ループ1回当りの平均所要時間は

$$44\alpha+150(1-\alpha)=150-106\alpha \quad \mu s$$

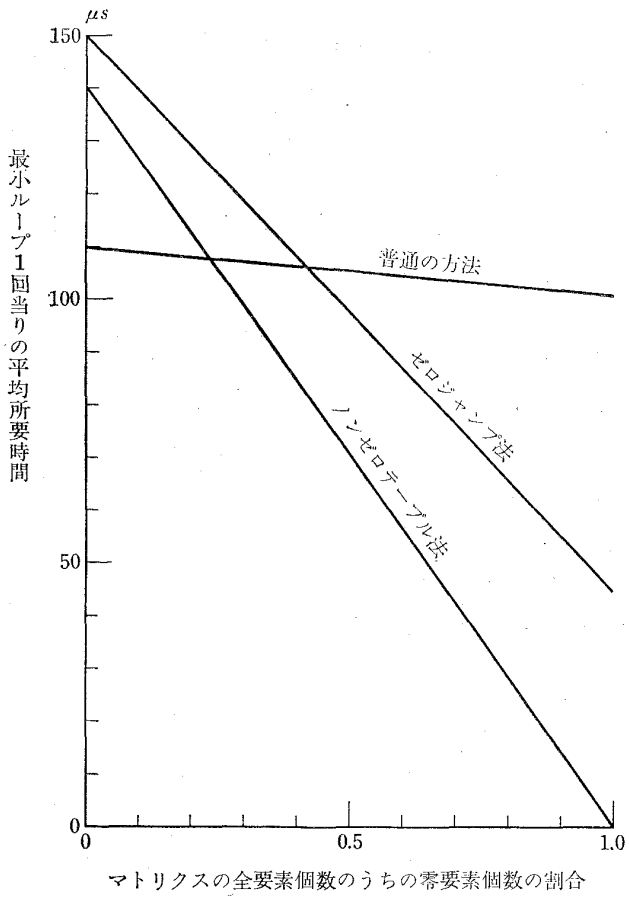

図1　ゼロジャンプ法およびノンゼロテーブル法の効果

となる。一方，もしもIFを入れなかったとすれば

$a_{ij}=0$ の場合　　106 μs

$a_{ij}\neq 0$ の場合　　110 μs

したがって平均所要時間は

$$106\alpha+110(1-\alpha)=110-4\alpha$$

となる。これを比較したのが図1である。この図から次のことがわかる。

(1) $\alpha>1/3$ ならばゼロジャンプ法が有利である。

(2) $\alpha\fallingdotseq 1$ すなわちマトリクスのほとんど全部が零の場合には計算時間は普通の方法の半分以下になる。

(3) $\alpha\fallingdotseq 0$ の場合は普通の方法よりも約30%の時間増となる。

ただし，以上の結論は HITAC 5020 の場合であって他の機種においてはかなり事情の異なる場合もある。

同じ手法は掃出法の計算にも応用できるが，掃出しのプロセスをくりかえしている間に非零要素が増加する傾向があるので，あらかじめ非零要素の配列が適切になるようにデータの入れ方を工夫するか，あるいは自動的に行，列の順序を調整して適切な形に整列する

ためのプログラムを準備する必要がある。

第2部のプログラムの大部分は，ゼロジャンプを行なうプログラムと，そうでないものとの二種類を用意してあるので，状況に応じて選択して使用することができる。

〔ノンゼロテーブル法〕 ゼロジャンプ法はプログラミングは簡単であるが，計算のたびに $a_{ij}=0$? のテストをするので計算時間短縮の効果が少なく，また零要素もデータとして記憶するので容量的問題の解決にはならない。この点を改善するため，零でない要素だけを記憶し処理するということを考える。このために，1個の非零要素 a_{ij} に対し，3語を用いて

行番号	列番号	値
i	j	a_{ij}

の形で表わす。これをすべての非零要素について集めて表にしたものを「ノンゼロテーブル」と呼ぶ。たとえば

$$\begin{pmatrix} 0 & 0 & 1 \\ 3 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

は，つぎのように表現される。

行番号	列番号	値
1	3	1
2	1	3
2	3	6

行番号，列番号，値に，たとえば

行番号	列番号	値
IAIJ	JAIJ	AIJ

という変数名をつけ，非零要素の個数をKとすれば，IAIJ，JAIJ，AIJ はいずれもディメンジョンKの添字付変数となる。これを用いて例1と同じ計算を行なうとすればつぎのようになる。

(例2)

```
      DO  1  I=1, N
1     Y(I)=0
      DO  2  L=1, K
      I = IAIJ(L)
      J = JAIJ(L)
2     Y(I)=Y(I)+AIJ(I,J)*X(J)
```

このループの1回当りの所要時間は142マイクロ秒で，やや長い。しかし零要素に関しては所要時間0であるから，マトリクス全体としてはループ1回当りの平均所要時間は

$$0\cdot\alpha+142(1-\alpha)=142-142\alpha \quad \mu s$$

となり，零要素の割合が大きい場合にはゼロジャンプ法よりはかるに有利になる。

次に記憶容量の点から見ると，N×Nマトリクスを収容するために普通の方法では N^2 語必要とするのに対しノンゼロテーブル法では3K語(Kは非零要素の個数)であるから，$\alpha>2/3$ ならば普通の方法より有利になり，$\alpha\sim 1$ の場合には非常に容量の節約になる。

実際の問題の場合に α の値がどの位になるかというと，工学上の問題の多くの例においてはマトリクスの一つの行に現れる非零要素の個数は，マトリクスの大きさに関係なくほぼ一定しており，たとえばマトリクス法による骨組構造解析の場合に12個，18個，24個というような数である。したがってマトリクスが大きければ非零要素の割合は小さくなり，$N>300$ 程度の問題においては非零要素は10%以下，すなわち $\alpha>0.9$ というケースが珍らしくない。そこで代表例として $\alpha=0.9$ の場合について評価してみると，計算時間は普通の方法の1/7以下（図1），記憶容量は普通の方法の1/3以下ということになる。

ただし，この方式で計算を行なうためには，データをあらかじめノンゼロテーブルの形で表現しておかなければならない。そのためには，普通の形式でストアされているマトリクスをノンゼロテーブルに書きかえるプログラム，およびその反対にノンゼロテーブルから普通の形に書きかえるプログラムが必要になる。またノンゼロテーブルを用いると記憶容量が節約されるため，普通ならばコアメモリに収容できないような大きなマトリクスも変換後にはコアメモリーに入れられる場合が少なくないので，磁気ドラムまたは磁気テープの上で組立てた大きなマトリクスの中から非零要素だけを拾ってコアに収容する，ということができると便利である。第2部の NZEMDC，NZEMTC などが，このような目的に使用するためのサブルーチンである。

第2部の LEAZ は ノンゼロテーブルを用いた掃出法による逆行列計算のプログラムである。また，反復法による連立一次方程式のノンゼロテーブルを用いたものとしては，NZCGSP，NZCGN がある。ユーティリティー・ルーティンとしては，NZMVM，TNZMVM (マトリクスとベクトルの積，転置されたマトリクスとベクトルの積を求める。)などがノンゼロテーブル方式によっている。

〔直接コーディング法〕 計算の種類によっては，マト

リクスの特定の位置に常に零が来る場合がある。非零要素の個数が少なければ，このような場合，一般的なDOループを使用せず，常に零になる部分を省いたプログラムを作れば，計算時間も記憶容量も非常に節約になる。鉄道技術研究所で開発された，車両の強度計算のプログラムは，この方法で非常に効果をあげているという[3]。

(例3)　方程式

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

において，a_{12}，a_{21}，a_{23}，a_{32}が常に零である場合，直接コーディングで書けば，

```
A13=A13/A11
B1 =B1/A11
X3 =(B3-A31*B1)/(A33-A31*A13)
X2 =B2/A22
X1 =B1-A13*×3
```

となる。A(1,3) というような添字を用いずにA13としたのは実行時間を短縮するためと記憶容量を節約するためである。上の例では，普通のDOループを用いた方法と比較して，計算時間は約60%減，メモリは約45%減（係数行列の部分に関し）となる。もっと大きな問題で零要素の割合が大きければ効果は上例よりはるかに大きくなる。

〔分割法〕　問題の種類によっては，マトリクスを適当な小行列に分割すると，特定の小行列が常に零行列になる場合がある。このような場合には，小行列を単位として，上に述べた直接コーディング法を適用することができる。この方法は分割法(Partitioning method)と呼ばれ，古くからよく知られている[5),6),7),8),9)]。記憶容量の効率も計算速度の効率も，前記のような1ワード単位の直接法に比較するといくぶん劣るが，非常に大きなマトリクスの計算を直接法で書くことはプログラムが長くなって事実上は不可能なので，分割法の方がよく用いられる。

(例4)　分割法の応用で最も重要な形は，block-tridiagonal form すなわち

$$M = \begin{pmatrix} A_1 & C_1 & & & & \\ B_2 & A_2 & C_2 & & & \\ & B_3 & A_3 & C_3 & & 0 \\ & & \cdots\cdots & & & \\ & & & \cdots\cdots & & \\ & 0 & & B_{m-1} & A_{m-1} & C_{m-1} \\ & & & & B_m & A_m \end{pmatrix}$$

の場合である。ただし A_i，B_i，C_i は小行列で A_iはすべて正方行列である。この場合に連立一次方程式 $Mx=y$ を解く掃出法の計算手順は，

$$A_1^{(1)}=\mathrm{I} \quad C_1^{(1)}=A_1^{-1}C_1 \quad y_1^{(1)}=A_1^{-1}y_1$$
$$B_2^{(1)}=0 \quad A_2^{(1)}=A_2-B_2C_1^{(1)} \quad y_2^{(1)}=y_2-B_2y_1^{(1)}$$
$$A_2^{(2)}=\mathrm{I} \quad C_2^{(2)}=A_2^{(1)^{-1}}C_2 \quad y_2^{(2)}=A_2^{(1)^{-1}}y_2^{(1)}$$
$$B_3^{(2)}=0 \quad A_3^{(2)}=A_3-B_3C_2^{(2)} \quad y_3^{(2)}=y_3-B_3y_2^{(2)}$$
$$A_3^{(3)}=\mathrm{I} \quad C_3^{(3)}=A_3^{(2)^{-1}}C_3 \quad y_3^{(3)}=A_3^{(2)^{-1}}y_3^{(2)}$$

以下同様で一般に（$k=n-1$ まで）

$$A_k^{(k)}=\mathrm{I} \quad C_k^{(k)}=\mathrm{A}_k^{(k-1)^{-1}}C_k \quad y_k^{(k)}=A_k^{(k-1)^{-1}}y_k^{(k-1)}$$
$$B_{k+1}^{(k)}=0 \quad A_{k+1}^{(k)}=\mathrm{A}_{k+1}-B_{k+1}C_k^{(k)} \quad y_{k+1}^{(k)}=y_{k+1}-B_{k+1}y_k^{(k)}$$

これより解は

$$x_m = \mathrm{A}_m^{(m-1)-1}y_m^{(m-1)}$$
$$x_{m-1} = y_{m-1}^{(m-1)} - C_{m-1}^{(m-1)}x_m$$
$$x_{m-2} = y_{m-2}^{(m-2)} - C_{m-2}^{(m-2)}x_{m-1}$$
$$\cdots\cdots\cdots\cdots$$
$$x_2 = y_2^{(2)} - C_2^{(2)}x_3$$
$$x_1 = y_1^{(1)} - C_1^{(1)}x_2$$

となる。（ ）内の添字は第k段の操作の結果を(k)で表わしている。たとえば$\mathrm{A}_k^{(k-1)}$は第$k-1$段終了後の A_k の値を表わす。$A_k^{(k-1)^{-1}}$はその逆行列である。( )の付かない項は最初の入力データそのままの値である。$A_k^{(k-1)^{-1}}C_k$，$y_k^{(k)}=\mathrm{A}_k^{(k-1)^{-1}}y_k^{(k-1)}$等の計算は，複数個の連立一次方程式を解くサブルーチン（小さなマトリクス用の）を用いて

$$A_k^{(k-1)}X = C_k$$
$$A_k^{(k-1)}x = y_k^{(k-1)}$$

を解いて求める。（後述するように，逆行列を求めてから行列乗算を行なうという手順は時間的に損である。）

mがあまり小さくなく，小行列 A_i，B_i，C_i のサイズがほぼ等しい場合には，計算量は一般の場合の約$6/m^2$，必要な記憶容量は約$3/m$で済む。たとえば $m=10$の場合，計算時間は6%，記憶容量は30%程度で済むことになる。

分割法を用いれば，かなり大きなマトリクスでも，コアメモリに収容できる程度の小行列に分割して処理できるので，この意味でも計算効率を良くする結果になる。骨組構造解析のプログラムとして有名なFRAN[2)]は分割法を巧みに使って，大きな効果をあげている。

1.3 外部記憶装置の能率的使用

大型計算機の演算速度は，以前の中型計算機の100倍以上になっているが，外部記憶装置の速度はそれほど速くなっていない。たとえば中型計算機の場合の磁気ドラムの平均待時間は，1演算命令ステップの実行時間とあまり変らない程度であったが，大型計算機の磁気ドラムの平均待時間は，演算命令数百ステップに相当する，したがって大型計算機の場合の，外部記憶装置の能率的使用の重要性は，中型計算機の場合よりもはるかに大きい，外部記憶装置を使用するために起る時間損失をできるだけ軽くするためには，つぎのような方法が基本的である。

(1) 外部記憶装置とやりとりするデータの量の総計をできるだけ少なくする。

(2) 外部記憶装置を呼び出す回数を，できるだけ少なくする（逆に言えば，1回でなるべく大量のデータを同時に転送するようにする)。

(3) データの配列を適切にして，むだな待時間を生じないようにする。

この理由は明白であろう。簡単に説明すれば，(1)は転送時間を短縮するため，(2)はデータ1個当りの待時間を減少させるため，(3)は待時間そのものを短縮するためである。

ただし，上記(1)は計算方式の全体的な問題であって，特別な解決技術はない，しいて言えば，先に述べたノンゼロテーブル法などを用いてメモリを節約すること，あるいは分割法を用いて計算の処理単位を適当な大きさにすること，などがこの目的に役立つ。

(2)のためには，〔**ブロック入出力**〕ということを行なう，これは外部記憶装置とのデータのやりとりを1語ずつにしないで何語かをまとめてブロックにして行なうことである。通常は，プログラムを簡単にするためマトリクスの1行分または1列分のデータを1ブロックとして扱うが，1ブロックの長さは（後に述べる理由から）1000語以上が理想的であって，このためには，200×200程度のマトリクスの場合には5行分あるいは5列分ぐらいのデータを1度に転送することが望ましい，仮に5行分のデータを1ブロックとする時，マトリクスの行数が5の倍数でない場合には最後に半端が出て，このためにプログラムはかなり複雑になる。しかし計算時間はこれによって著しく改善されるので，少々苦労してもブロックサイズの大きいプログラムを書くべきである。

(例5) 300元の連立一次方程式 $Ax=b$ を掃出法で解く，データが全部コアメモリに入っているとすれば計算時間は約15分であるが，実際にはコアメモリに入りきらないので磁気ドラムに入れることになる。まず，磁気ドラムから1語ずつ取り出し1語ずつ書き込む場合を考えると，磁気ドラムを呼び出す周波は約18,000,000回で待時間が1回当り約20ms かかるので，全部で

$$20\text{ms}\times 18,000,000=360,000\text{s}=100\text{時間}$$

を要することになる，次に，1行分300語を1ブロックとして転送する場合を考えると，磁気ドラムの呼び出し回数は約45,000回，1回当りの所要時間（待時間，転送時間，計算時間を含む）は平均60ms 程度なので

$$60\text{ms}\times 45,000=2,700,000\text{ms}=45\text{分}$$

となる。次に，1,000語を1ブロック（すなわち5行分で1ブロック）とすれば全所要時間は約18分ぐらいになる。以上の結果をまとめると，この例の場合，外部記憶装置使用のための時間（待時間等）の割合は，

1ブロック1語のとき	全所要時間の99.7%
1ブロック200語のとき	全所要時間の 67%
1ブロック1000語のとき	全所要時間の 17%

となる。計算時間の何倍もの時間を待時間に費すことは不経済なことであるから，できるだけ大きいブロックにするよう努力すべきであろう。

改善案(3)の，データの配列を工夫する，ということは磁気テープを用いる場合に特に重要である，巻き戻しや back space の時間をなるべく少なくするため，磁気テープ装置を複数台使用する方式がよく用いられる，掃出法の場合は（他の理由もあるので）必らず2台以上用いて1台から $a_{ij}^{(k)}$ を読み他のテープに $a_{ij}^{(k+1)}$ を書き込むようにするべきである，理想的には，テープを3台使用して，図2のU，処理前のA，処理後のAに各1台を割当てれば，転送データ量もしだいに少なくなるので，良い結果が得られる。反復法の場合には，マトリクスAを記録した同じテープを2本作って1本で計算をしている間に他の1本を巻き戻すようにすれば時間の短縮に効果がある（ただし当所のシステムを FORTRAN で使用している場合には巻き戻しと計算の同時処理ができないので，この方法は利用できない。)。

磁気ドラムの場合，いわゆる optimum coding* をすれば効果が大きい。マトリクス関係の一般的サブルーチンに応用することは困難であるが，サイズの固定した問題にはある程度，応用できるかも知れない，当所で現在使用している H-179A 型 磁気ドラム装置の

* 1段階の計算が終了した時に，次の段階に必要なデータがちょうど読み取りヘッドの所に来るように，磁気ドラム上のデータを配列すること。

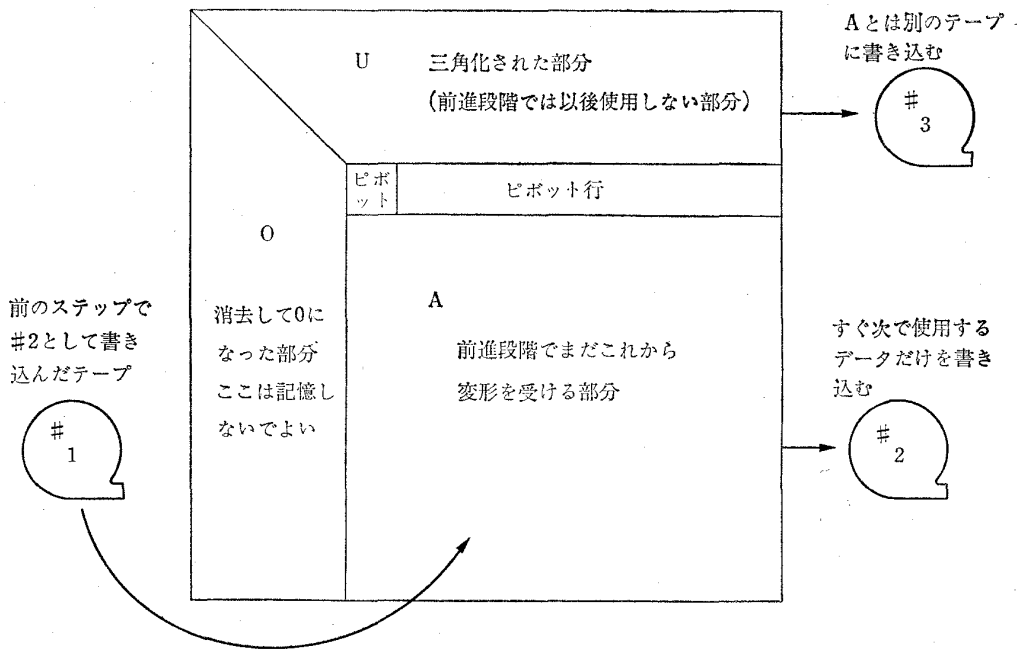

図 2　磁気テープを 3 台使用した掃出法前進計算

主要諸元はつぎのとおりである。

1 回転の所要時間	20ms
バンド数（1 台当り）	64
1 バンド当りの語数	1,024語
読取り書込みのヘッドの配置	1 周に 1 個所
転送速度	約20 μs/語
番地付けの順序	図 3 参照

optimum coding の際には，上記のハードウェア特性のほかに，磁気ドラム入出力制御ルーチンを通過する時間（ごくわずかであるが）を考慮すべきである。たとえば

WRITE D(1), (A(I), I=1,100)
READ D(101), (B(I), I=1,100)

の場合，WRITE の終了時点には D(101) 以降のデータをただちに読める位置にヘッドがあるはずであるが，入出力制御ルーチンの関係その他の理由から，READ の実行までにわずかの時間遅れが生じ，このため実際には D(101) はヘッドの所を過ぎてしまうので次にこれが回って来るまでドラム 1 回転分 20ms 待たされることになる。

HITAC 5020 の場合には，以上のほかに，

READ DRUM PROCEED
WRITE DRUM PROCEED

なるステートメントを用いて時間を短縮することができる。これを用いれば待時間と転送時間の間に計算を実行するので適切な使用法をすれば，かなりの効果を

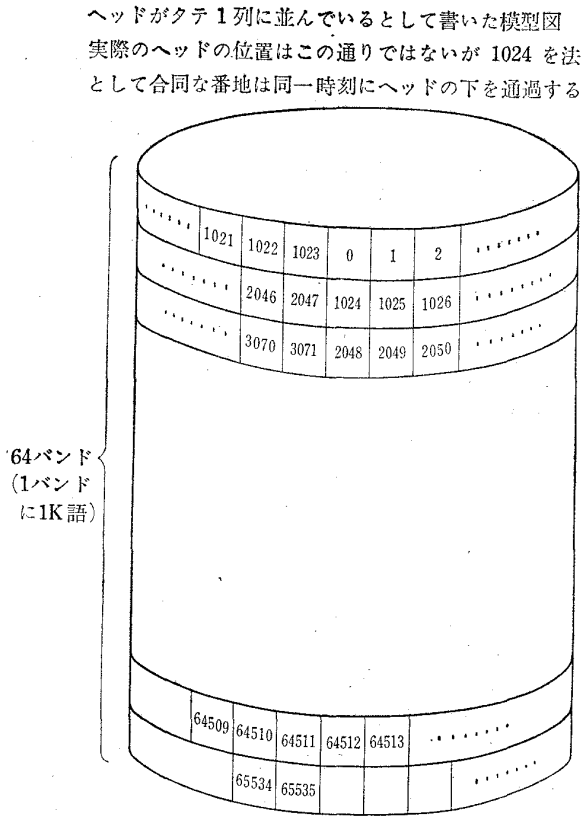

図 3　磁気ドラム概念図

あげることができる。しかし計算時間に比較して待時間の方が長い場合には全く効果がないので，これを避けるためには転送のブロックを大きく（大体の目安としては300語以上に）するか，あるいはデータの配列を optimum coding に近い形にするなどの措置が必要である。なお磁気テープに関しては，FORTRAN の PROCEED ステートメントは無く，HISAP によっても（まともな方法では）書けない。

磁気テープを FORTRAN で使用する場合，1個の WRITE ステートメントで書き込むデータの量が256語以下の場合には，自動的に空白のデータが追加されて1ブロック256語にして書き込む方式になっているので，あまり小さな単位で磁気テープ転送を行なうと時間的に損になる。

1.4 機械語の使用

プログラムを FORTRAN で書くと，機械語（またはアセンブラ言語）で書く場合に比較して，計算の時間が少し長くなる。その原因は，コンパイラの処理が機械的，画一的であるため，あるいは FORTRAN の文法が効率の良いプログラムを書けないようになっているため，その他いろいろあるが，結果としては1～2割程度の時間増となる。

HITAC 5020 の場合，マトリクス関係の計算で，FORTRAN を使用したために時間損失を生じる主な原因はつぎのようなものである。

(1) A(I, J)のように添字を2個使用すると，MNI (Modify Next Instruction) 命令で修飾されるため1回当り 20μs 増となる。機械語でコーディングして命令の mパートを変更するようにすれば，この時間増は除去できる。

(2) マトリクスの積，ベクトルの内積の計算などの場合，S＝0としておいて

```
DO  1  I=1,N
1   S=S+X(I)*Y(I)
```

とするとSを1回ごとにコアメモリにストアするために，1回当り16μsの時間を要する。機械語で書けばSはレジスタに残しておくので，この時間が節約される。

(3) ゼロジャンプ法の場合，FORTRAN ならば

```
IF(A(I,J).EQ.0.)
```

という独立した判別操作をしなければならないが，機械語ならばJZ (Jump on Zero) 命令1個を挿入するだけで済み，FORTRANより12μs速くなる。

積和計算，掃出計算等の最小ループの所要時間は 100 μs前後であるから，上記の処置で約1～2割スピードアップされることになる，

1.5 計算精度の保持と改良

〔倍長語計算〕 von Neumann および Goldstineの研究[10]によれば，掃出法により逆行列を求める場合，ほとんどすべての問題に対し（正確に言えば，random matrix の99%に関して）

$$|A\tilde{A}^{-1}-I|<1$$

（ただし A は元のマトリクス $\tilde{A}^{-1}$ は計算で得られた逆行列，I は単位行列）の程度の精度を保証するためには，計算中の有効数字を

$$3\log_2(n/0.04)\ \text{ビット}$$

より長くとっておけば安全であるという結論が出ている。これで保証できる精度は「オーダーは合っている」という程度であるが，有効数字をさらにSビット長くとれば，結果も（厳密に言えばマトリクスAの性質にもよるが）約Sビットまで正確になることが期待できる。

HITAC 5020 の場合，浮動小数点単長語の有効数字の長さは23ビットであるから，上の理論に従うならば，これで解けるのはせいぜい10元どまりで，これ以上は倍長語計算が必要になる。倍長語ならば有効数字の長さは55ビットあるので，400元でも18ビットの余裕があり，3桁程度の精度が期待できる。

上の議論は掃出法の場合であるが，反復法を用いる場合はそれほど有効数字を長くとる必要はなく，かなり大きなマトリクスでも単長語で比較的よい結果が得られる場合が多く，ある程度の近似値が得られてから高精度計算に切換えて計算することも可能である。しかし反復法の場合には，倍長語で計算するとしても，そのためのメモリ必要量の増加はごくわずかであり，HITAC 5020 の場合には計算時間もあまり変らないので，かなり精密な結果が要求される場合には倍長語にしておく方がよいであろう。特にCG法（後述）の場合には丸めの誤差が後の計算に悪い影響を残すので倍長語で扱っておくのが安全である。

次に精度の改善の方法であるが，これは連立一次方程式の場合と逆行列の場合とで異なる。連立一次方程式の場合には適当な反復法を適用するのが最も簡単で時間も短かく有効である。逆行列の場合には下記の方法が最も有効である。(4) A の逆行列 A^{-1} は，方程式

$$AX=I$$

の解 X である。その近似値(第1回の計算で得られた結果)を $\tilde{A}^{-1}$ とするとき，上式の両辺に $\tilde{A}^{-1}$ を掛けて

$$(\tilde{A}^{-1}\ A)X=\tilde{A}^{-1}$$

なる方程式を作ると，係数行列 $\tilde{A}^{-1}A$ は単位行列に近い値になるはずであるから，この方程式は有効数字の桁落ちなどがあまり起らない解きやすい形（いわゆる well-conditioned equation）になっているので，$\tilde{A}^{-1}$ よりも良い近似値を得ることができる。著者らの経験によれば，これを数回くり返すことにより非常に良い結果を得ることができた（実例　26ページ参照）。

1.6　解法の選択について

逆行列計算および連立一次方程式の計算の計算法（数値解法）の種類は非常に多く，個々の計算法の性質について数多くの研究が発表されているが，これら全体を論ずることは本資料の目的ではないので，実用上特に重要な事項だけを説明しておく。

連立一次方程式の解法は大別して

- 直接法
- 間接法（反復法）

の二種類がある。この区分の厳密な定義は困難であるが，常識的には，「計算中に誤差が全く混入しなければ有限回のステップで厳密解に到達するものが直接法」，「粗い近似値から出発して，反復によりしだいに精度を高めていくのが間接法」と考えてよいであろう。

直接法に属する解法としては

- 掃出法（消去法）
- 直交化法
- エスカレータ法
- クラメルの公式
- その他

がある。また掃出法の中にも消去の順序によっていくつかの公式がある。直接法は，解が得られるまでに要する計算量（普通は乗算回数で代表させる）が公式ごとに確定しているから，精度の点を一応あとまわしにして計算時間だけに注目すれば解法の比較は簡単にできる。各解法の乗算回数は，細部の手順のとりかたによって若干の差はあるが，だいたいのオーダーはつぎのとおりである。

掃出法	Gauss 法	$n^3/3$ 回
	Cholesky 法	$n^3/3$ 回
	その他	$n^3/2 \sim n^3$ 回
直交化法		$n^3 \sim (5/3)n^3$回
エスカレータ法		$2n^3$回以上
クラメルの公式		n！に比例するので論外

また，精度に関していろいろな比較研究の結果が発表されているが，計算時間の長い解法が必らずしも精度が良好ではなく，あまり差のないことが認められている。そこで，乗算回数の最も少ない Gauss法と Cholesky法を比較してみると，高精度計算の場合に Cholesky 法ならばワーキング・スペース1個分だけを倍長語（あるいはそれ以上の高精度語）にするだけでよいが，Gauss 法の場合には n^2 個の高精度語をストアする必要があり，この点では Gauss法が不利であるが磁気ドラムまたは磁気テープを使用する場合には，常に行単位で（または常に列を単位として）単純な順序で処理できる Gauss法が，時間的にもプログラムの単純さの点からも非常に有利になる。本資料では，磁気ドラム，磁気テープの関係が多いため，Gauss 法を用いた。その公式は，簡単に説明するとつぎのとおりである。

公式1（掃出法，Gauss方式）

第1行を a_{11} で割る。

第2行から，上の結果の a_{21} 倍を引く。

第3行以下も同様にして第1列を消去する。

第1行，第1列を除いた部分について同様処理。

以下同様の反復で対角より左下部を零にする。

これが完了したら第 n－1 行から順に逆代入。

なお詳細については，数値計算法の教科書あるいは文献[1]を参照されたい。

つぎに，間接法に属する解法としては，連立一次方程式に関しては代表的なものとして

ＧＳ法（Gauss-Seidel 法）

ＳＯＲ法（Successive Over-relaxation 法）

ＣＧ法（Conjugate Gradient 法）

ＳＤ法（Steepest Descent 法）

がよく使用されている。このほかにも多くの反復公式が発表されているが上記のものの方が概して収束が速く使いやすい。

反復法を使用する場合には，その公式の適用できる範囲に注意する必要がある。不適当な問題に適用すると収束せず反対に発散することがある。適用範囲のせまい公式は，適用範囲の広い公式に比較して概して収束が速い。たとえば以下の公式3と公式4を比較すると，正値対称の問題に関しては公式3の方が収束率が良く，また反復1回当りの計算時間も短かい。

第2部のサブルーチンに用いた計算公式を以下に記す。

公式2（ＧＳ法）

各式の両辺を対角要素の値で割って，あらかじめ対角要素を1にしておく。$k=1, 2$……に対し

$$x_1^{(k+1)} = c_1 - (\quad + a_{12}x_2^{(k)} + a_{13}x_3^{(k)} + \cdots + a_{1n}x_n^{(k)})$$

$$x_2^{(k+1)} = c_2 - (a_{21}x_1^{(k+1)} + \quad + a_{23}x_3^{(k)} + \cdots + a_{2n}x_n^{(k)})$$

$$x_3^{(k+1)} = c_3 - (a_{31}x_1^{(k+1)} + a_{21}x_2^{(k+1)} \quad \cdots + a_{3n}x_n^{(k)})$$

$$\cdots\cdots$$

$$\cdots\cdots$$

$$x_{n-1}^{(k+1)} = c_{n-1} - (a_{n-1,1}x_1^{(k+1)} + \cdots + a_{n-1,n-2}x_{n-2}^{(k+1)} + a_{n-1,n}x_n^{(k)+1})$$

$$x_n^{(k+1)} = c_n \quad - (a_{n1} \quad x_1^{(k+1)} + \cdots + a_{n,n-2} \quad x_{n-2}^{(k+1)} + a_{n,n-1}x_{n-1}^{(k+1)} + \quad)$$

ただし，$c = (c_1, c_2, \cdots\cdots, c_n)$ は方程式の右辺，$A = (a_{ij})$ は係数行列，$x^{(k)} = (x_1^{(k)}, x_2^{(k)}, \cdots\cdots x_n^{(k)})$ は第 k 近似解。

公式3 （CG法）

方程式を $Ax = C$ とし，その第0近似解を $x^{(0)}$ とする。この $x^{(0)}$ と，

$$p^{(0)} = r^{(0)} = C - Ax^{(0)}$$

から出発して，つぎの反復で逐次 $x^{(1)}$, $x^{(2)}$, …… を求める。

$$\alpha^{(k)} = (r^{(k)}, r^{(k)}) / (Ap^{(k)}, p^{(k)})$$
$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} p^{(k)}$$
$$r^{(k+1)} = r^{(k)} - \alpha^{(k)} A p^{(k)}$$
$$\beta^{(k)} = (Ap^{(k)}, r^{(k+1)}) / (Ap^{(k)}, p^{(k)})$$
$$p^{(k+1)} = r^{(k+1)} - \beta^{(k)} p^{(k)}$$

ただし（，）は内積，小文字の変数はベクトル，ギリシャ文字はスカラー

公式4 （CG法，一般行列用）

上記の公式3はCG法の原型で最もよく知られているものであるが，係数行列Aが正値対称でないと適用できない，という限界がある。そこで正値対称でない一般の問題に適用できるよう変形した公式がいくつか提案されているが，つぎのものはその一例である。記号は3と同じで

出発値

$x^{(0)}$……適当に与える

$$r^{(0)} = C - Ax^{(0)}$$
$$p^{(0)} = A^T r^{(0)}$$

反復式

$$\alpha^{(k)} = (A^T r^{(k)}, A^T r^{(k)}) / (Ap^{(k)}, Ap^{(k)})$$
$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} p^{(k)}$$
$$r^{(k+1)} = r^{(k)} - \alpha^{(k)} A p^{(k)}$$
$$\beta^{(k)} = (A^T r^{(k+1)}, A^T r^{(k+1)}) / (A^T r^{(k)}, A^T r^{(k)})$$
$$p^{(k+1)} = A^T r^{(k+1)} + \beta^{(k)} p^{(k)}$$

ただし A^T は A の転置行列。

公式5 （SD法）

$$r^{(k)} = Ax^{(k)} - C$$
$$\alpha^{(k)} = (r^{(k)}, r^{(k)}) / (r^{(k)}, Ar^{(k)})$$
$$x^{(k+1)} = x^{(k)} - \alpha^{(k)} r^{(k)}$$

つぎに逆行列計算であるが，これも直接法と間接法とがあり，直接法としては掃出法が，また間接法としては反復式

$$X^{(k+1)} = X^{(k)} (2I - AX^{(k)})$$

（ただし，$X^{(k)}$ は A^{-1} の第 k 近似行列）

などがよく用いられているが，掃出法によれば n^3 回の演算で逆行列が得られるのに対し，反復法を用いると一回の反復だけでも $2n^3$ 回の演算が必要になるので，（係数行列に零要素が非常に多い場合を除けば）掃出法を用いる方が概して有利である。

公式6 （掃出法による逆行列計算）

右辺に単位行列を置いて公式1と同じ操作を行なう。

詳細は文献[5)1)]などを参照されたい。

第2部 応用サブルーチン

〔I〕 連立一次方程式

先に1.6項で説明したように，連立一次方程式の大きな問題においては反復法が有利であり，この理由からここでは反復法のサブルーチンだけを収録した。

使い分けは

係数に零が多ければ ⟶ 2.2

係数に零が多くない場合で

- 50元までならば ⟶ 2.1
- 200元までならば ⟶ 2.3
- それ以上は ⟶ 2.4

また正値対称ならば必ず正値対称専用のサブルーチンを使用すべきである（計算時間が短かい）。

同一の係数行列で右辺のみ異なる連立方程式を何ケースも解く場合は，計算をかなり省略できることがあるので下記の方針で使用されたい。

係数に零が多く

- ケースの数が少なければ個々に2.2を適用
- block-diagonal 等の場合⟶ 2.8
- その他は「零が多くない場合」にずる

係数に零が多くない 場合で

- 50元までならば ⟶LINSWN[1)]
- 100元以下でケースの数が多ければ
  逆行列を計算して右辺の行列に掛ける
- 200元まで ⟶ 2.7備考

ケースの数が少ない場合には，2.1, 2.3, 2.4を個々の

ケースに適用してもよい

2.1　50元までの方程式を解く

50元までならば既製のサブルーチン[1]でも処理できるが，以下の4個のサブルーチンは既製のものよりも計算時間の点で大幅に改善されている。

こちらの方が速い理由は，

i）主要部分（行列をベクトルに掛ける部分）を機械語（HISAP）でコーディングしている。

ii）係数行列に零が多い場合には，ゼロジャンプ法が使用できる。

iii）正値対称の問題の場合には，収束の速い解法を使用できる，

などである，

次数制限は一応50元までとなっているが，サブルーチンの中の DIMENSION ステートメントを書き換えて，ワーキングスペース XX, R, P, Y, を必要次数まで拡大すれば，コアメモリに収容できる範囲内で，大きな問題を扱うことができる。たとえば100元を解きたい場合は

```
DIMENSION XX(100),R(100),P(100),Y(100),
A(N,N),X(N),C(N)
```

とすればよい

〔用語の説明その他の注意〕

(1) 引数の説明で「実行前」の欄に書かれているものは，サブルーチンを呼び出す前にすべて与えておかなければならない。

(2) 最大反復回数で指定した回数まで反復して，それでも指定精度が得られない場合は，そのままメインプログラムに帰る。

(3) 「許容誤差」の意味は，第k近似値と第$k+1$近似値の差を一応，誤差の目安とし，これが許容誤差範囲内におさまるまで反復をくりかえす。

(4) GS法の収束条件は，十分条件として，正値対称であること，または各行について

$$2a^2_{ii} > \sum_{j=1}^{n} a^2_{ij}$$

が成立すること。

(5) 第1近似値を特に指定しなければ，右辺Cを第1近似として出発する。

CGSP

目的

連立一次方程式 $Ax=C$ を解く

解法

CG法（公式3）

適用条件

係数行列が正値対称であること。50元まで。

呼び方

```
CALL CGSP (A, C, X, N,M,I,E)
```

引数

引数名	実行前	実行後	次元	型
A	係数行列A	(不変)	N×N	実
C	右辺C	(不変)	N	実
X	第1近似解 (I＝0の時は関係なし)	解 x	N	実
N	次数	(不変)	—	整
M	最大反復回数	反復回数	—	整
I	第1近似解の指定の有無 1有，0無	(不変)	—	整
E	許容誤差	(不変)	—	実

使用サブルーチン

TMAVEM, VECMUL

SDSP

目的

連立一次方程式 $Ax=C$ を解く

解法

SD法（公式5）

適用条件

係数行列が正値対称であること。50元まで。

呼び方

```
CALL SDSP (A, C, X, N, M, I, E)
```

引数

CGSP に同じ（11頁参照）

使用サブルーチン

TMAVEM, VECMUL

GSP

目的

連立一次方程式 $Ax=C$ を解く

解法

GS法（公式2）

適用条件

GS法の収束条件（11頁）参照。50元まで。

呼び方

```
CALL GSP (A, C, X, N, M, I, E)
```

引数

CGSP に同じ（11頁参照）

注意

係数行列の対角要素は，すべて1にしておかなければいけない。対角要素が1でない場合は，あらかじめ MADIA を用いて両辺を常数で割っておくこと。

使用サブルーチン

なし

C G N

目的

連立一次方程式 $Ax=C$ を解く

解法

一般行列用のCG法（公式4）

適用条件

50元まで

呼び方

CALL CGN (A, C, X, N, M, I, E)

引数

CGSP に同じ (11頁参照)

使用サブルーチン

TMAVEM, MAMULT, VECMUL

2.2 係数に零が非常に多い場合を解く

係数行列の中の零要素の個数が，零でない要素の個数よりはるかに多い場合に適用するためのサブルーチンで，先に1.2で説明したノンゼロテーブル法を用いて記憶容量および計算時間の節減を計っている。

次数に関する制限はなく，ノンゼロテーブルがコアメモリーに収容できる限り，いくらでも大きい問題が扱える。通常の表現ではコアメモリに収容できないような数百元の問題に適用すると非常に有効である。ノンゼロテーブルの作成は，各自のプログラムで行なってもよいが，ASMTD (20頁) を用いて磁気ドラム内にマトリクスを組み立てて，しかる後に NZEMDC (24頁)によりノンゼロテーブルに変換すると簡単である。

N Z C G S P

目的

係数行列に零要素を多く含む連立一次方程式 $Ax=C$ を解く

解法

CG法（公式3）

係数行列の表現にノンゼロテーブルを使用

適用条件

係数行列が正値対称であること

次数制限なし

呼び方

CALL NZCGSP (AIJ, IAIJ, JAIJ, K, C, X, N, M, I, E, XX, R, P, Y)

引数

引数名	実行前	実行後	次元	型
AIJ	係数行列A	(不変)	K	実
IAIJ	行番号	(不変)	K	整
JAIJ	列番号	(不変)	K	整
K	非零要素個数	(不変)	—	整
C	右辺C	(不変)	N	実
X	第1近似解 (I＝0の時は関係なし)	解 x	N	実
N	次数	(不変)	—	整
M	最大反復回数	反復回数	—	整
I	第1近似解の指定の有無 1有，0無	(不変)	—	整
E	許容誤差	(不変)	—	実
XX	(無関係)	(破壊される)	N	実
R	(無関係)	(破壊される)	N	実
P	(無関係)	(破壊される)	N	実
Y	(無関係)	(破壊される)	N	実

なお，AIJ, IAIJ, JAIJ の意味，使用法に関しては第1部のノンゼロテーブル法の項（4頁）参照。

使用サブルーチン

VECMUL, NZMVM

備考

XX, R, P, Y はデータの受渡しには関係ないが，それぞれサイズNのディメンジョンを切っておかなければいけない。本来は，このサブルーチン専用のワーキングスペースなのであるが，同じスペースをサブルーチンの外でも活用できるように引数に入れてある。

N Z C G N

目的

係数行列に零要素を多く含む連立一次方程式 $Ax=C$ を解く

解法

一般行列用のCG法（公式4）

適用条件

制限なし

呼び方

CALL NZCGN (AIJ, IAIJ, JAIJ, K, C, X, N, M, I, E, XX, R, P, Y)

引数

NZCGSP に同じ（12頁参照）

使用サブルーチン

NZMVM, TNZMVM, VECMUL

2.3 磁気ドラムを使用して200元までの方程式を解く

係数行列がコアメモリに収容できないような大きな問題を解くためのサブルーチンである。

次数制限が一応200となっているが，下記の2点を書き換えれば，ドラムの容量の限度いっぱいまでの大きな問題を扱うことができる。

i） DRUM COMMON で宣言する領域を拡大する。

ii） XX, R, P, Y の dimension を拡大する，

たとえば300元の場合

DRUM COMMON A (90000)

DIMENSION XX (300)，R(300)，P(300)，Y(300)，C(N)，X(N)

係数に零が多い場合は2.2のサブルーチンの方が断然速いのでそちらを用いることを推奨する。

CGSPD

目的

連立一次方程式 $Ax=C$ を解く

解法

CG法（公式3）

適用条件

係数行列が正値対称であること，200元まで，

呼び方

CALL CGSPD (C, X, N, M, I, E)

引数

引数名	実行前	実行後	次元	型
C	右辺C	(不変)	N	実
X	第1近似解（I＝0の時は関係なし）	解 x	N	実
N	次数	(不変)	—	整
M	最大反復回数	反復回数	—	整
I	第1近似解の指定の有無 1有，0無	(不変)	—	整
E	許容誤差	(不変)	—	実

係数行列Aは

DRUM COMMON A (40000)

で引き渡す。

ドラムの中の配列は

$a_{ij}=A\ ((j-1)N+i)$

ドラムの中に係数行列を組み立てる際に，後述のユーティリティ・ルーチン

CLEARD, ASMTD

などを用いると便利である。

使用サブルーチン

MDMTVT, VECMUL

CGND

目的

連立一次方程式 $Ax=C$ を解く

解法

一般行列のためのCG法（公式4）

適用条件

200元まで

呼び方

CALL CGND (C, X, N, M, I, E)

引数

CGSPD に同じ （13頁参照）

使用サブルーチン

MDMTVT, MDMTV, VECMUL

2.4 磁気テープを使用して1000元ぐらいまでの方程式を解く

係数行列が磁気ドラムにも収容できないような大きな問題のためのサブルーチンである。

サブルーチンとしての次数制限は無いが，1巻の磁気テープに収容できる語数により制限を受け

記録密度　800 BPI*の時　1,600元

記録密度　200 BPI の時　1,000元

が実際上の上限である。（ただし，個々のテープにより長さのバラツキ，傷の有無などにより多少の差がある。） 記録密度 800 BPI で使用する場合はその旨を操作員に指示しておかなければいけない。

係数に零が多い場合は2.2のサブルーチンの方が断然速いのでそちらを用いることを推奨する。

CGSPT

目的

連立一次方程式 $Ax=C$ を解く

解法

CG法（公式3）

適用条件

係数行列が正値対称であること。

* BPI は bit per inch の略，長さ方向2.54ミリ当りのビット数，

呼び方

CALL CGSPT (C, X, N, M, I, E, K, XX, P, R, Y, W)

引数

引数名	実行前	実行後	次元	型
C	右辺C	(不変)	N×N	実
X	第1近似解 (I＝0の時は関係なし)	解 x	N	実
N	次数	(不変)	—	整
M	最大反復回数	反復した回数	—	整
I	第1近似解の指定の有無 1有，0無	(不変)	—	整
E	許容誤差	(不変)	—	実
K	磁気テープのユニット番号	(不変)	—	整
XX	(無関係)	(破壊される)	N	実
P	(無関係)	(破壊される)	N	実
R	(無関係)	(破壊される)	N	実
Y	(無関係)	(破壊される)	N	実
W	(無関係)	(破壊される)	N	実

係数行列は磁気テープで引き渡す。磁気テープの書込み方は，最初まず REWIND をかけ，次に，第1行，第2行，……，第N行の順に，必らず1行につき1個の WRITE ステートメントで記録する。

使用サブルーチン

VECMUL, MTMTVT

CGNT

目的

連立一次方程式 $Ax=C$ を解く

解法

一般行列用のCG法（公式4）

適用条件

制限なし

呼び方

CALL CGNT (C, X, N, M, I, E, K, XX, P, R, Y, W)

引数

CGSPT に同じ（13頁参照）

使用サブルーチン

VECMUL, MTMTV, MTMTVT

〔II〕 逆行列

先に1.6項で説明したように，逆行列計算に関しては掃出法が有利なので，掃出法によるサブルーチンだけを収録した。

2.5 100×100 まで

MAINV1

目的

行列Aの逆行列を求める

解法

掃出法 (Gauss-Jordan 法，公式6)

適用条件

100×100まで

呼び方

CALL MAINV1 (A, N, D)

引数

引数名	実行前	実行後	次元	型
A	行列A	逆行列A^{-1}	N×N	実
N	次数	(不変)	—	整
D	(無関係)	行列式の値	—	実

使用サブルーチン

なし

備考

オーバーフローを避けるため，このルーチンに入る前に NORMA (19頁) を用いて行列Aをノーマライズしておくことが望ましい。

MAINV2

目的

行列Aの逆行列A^{-1}を求める

解法

掃出法 (Gauss-Jordan 法，公式6)

また，下記の要領でピボット選択を行なっている。いま第p－1行まで処理済みとするとき，

i) 第p行から先の各行を

$$\alpha_i=\sqrt{\sum_{j=p}^{n}a_{ij}^{(p-1)^2}}$$

で割る（第i行の各要素（右辺を含めて）をα_iで割る）。

ii) $p\leqq j$, $p\leqq i$のすべてのi, jについて$a_{ij}^{(p-1)}$の絶対値を比較し，最大のものをピボットにする（このため，行，列の入れかえを行ない，最終結果が得られた後に復元する）。

適用条件

100×100まで

呼び方

CALL MAINV2 (A, N, D)

引数

MAINV1 に同じ

使用サブルーチン

なし

備考

(MAINV1 の項の備考参照)

2.6 精密計算

問題によっては，掃出法で普通に解いただけでは十分な精度が得られないことがある。このような場合にそなえて，掃出法計算ののち検算を行ない，精度が不十分であれば先に1.5項で説明した方法により精度改善を行なうプログラムを作成した。

既製のサブルーチン[1]にも精密計算と称するものがあるが，既製のものでは補正計算の際に行列の条件数(condition number) が全く改善されていないのに対し，本方式では1回ごとに条件数が飛躍的に改良されてゆくので，少ない反復で良い結果が得られる。

次数制限 ($N \leqq 50$) は，サブルーチンの中のDIMENSION ステートメント

A(N1, N1), Y(50, 50), S(50, 50), X(50, 50)

を書きかえれば緩和することができるが，全部で $4n^2$ 語のスペースを必要とする点に注意されたい。

MAINV3

目的

行列Aの逆行列A^{-1} を求める

解法

掃出法と$\tilde{A}^{-1}AX=\tilde{A}^{-1}$ の方法 (1.5項参照) により精密計算

適用条件

50×50まで

呼び方

CALL MAINV3 (A, N, D, E)

引数

引数名	実行前	実行後	次元	型
A	行列A	逆行列A^{-1}	N×N	実
N	次数	(不要)	—	整
D	(無関係)	行列式の値	—	実
E	許容誤差	(不変)	—	実

使用サブルーチン

MAINV2

備考

(MAINV1 の備考参照)

2.7 磁気ドラムを使用して 200×200 までの逆行列を求める

MAINVD

目的

行列Aの逆行列A^{-1} を求める

解法

掃出法 (Gauss-Jordan 法，公式6)

適用条件

200×200 まで

呼び方

MAINVD (M, N, B)

引数

引数名	実行前	実行後	次元	型
M	(=N)	(不変)	—	整
N	次数	(不変)	—	整
B	Aの第1行	破壊される	M	実

行列Aは DRUM COMMON で引き渡す。(CGSPD参照　13ページ)

使用サブルーチン

SMATIN (これは汎用ルーチンではなく，便宜上，一部をサブルーチン形式にしてあるだけであるが必ず付けること。)

備考

このサブルーチンは逆行計算ばかりではなく，連立一次方程式の計算にも使用することができる。特に係数行列が共通で右辺だけを変えて複数個の方程式を解く場合にも適用可能で，これを1回の掃出計算で処理できるので有利である。

連立一次方程式を解く場合は，行列Aの第N＋1列に（複数個の計算を同時に行なう場合は第N＋2列以下にも）右辺を入れ，

M　列の数（Nプラス式の数－1）

N　行の数

を引数として MAINVD をコールすると，結果の第N＋1列に(複数個の場合は第N＋2列以下にも)解が得られる。第N列まではA^{-1} である。

2.8 零要素が多い場合の逆行列計算または連立一次方程式の計算

標記の問題に関しては，先に〔Ⅰ〕の冒頭で述べたとおり，一般には2.2あるいは2.5，2.7等の方法が有利であって，以下で説明するＬＥＡＺが有効に使えるの

はマトリクスが block-diagonal の形をしている場合その他の特殊な場合に限られる。

ＬＥＡＺは，掃出法にノンゼロテーブル法を応用したプログラムである。ノンゼロテーブル法は，反復法の場合には，簡単に応用できるが，掃出法の場合には（特にノンゼロテーブルを部分的にでなく全面的に適用しようとすると）論理的に複雑になり，プログラミングが非常にむつかしくなる。この困難を冒して掃出法へのノンゼロテーブル法の応用を試み，その計算処理方式を確立しようというのが LEAZ を開発した主要な動機であって，実用性よりむしろ研究的性格を主体にしたものである。

プログラムの概要を説明すると，解くべき方程式を

$$AX=B$$

（ただしＡはｎ行ｎ列，ＸとＢはｎ行ｍ列のマトリクスで，Ｘについて解く。特にＢが単位行列ならば逆行列が計算され，$m=1$ ならば普通の連立一次方程式となる）とするとき，マトリクスＡ，Ｂ，Ｘをノンゼロテーブルで表現するのはもちろすであるが，そのほかに計算の中間結果として現れる各種のベクトルやサブマトリクスも，すべてノンゼロテーブルで表現している。この中間値のためのテーブルとしては，つぎのようなものがある。掃出法の前進段階の計算中の状態は図５のように分けられる。左下の０の部分は消去されて０になった所で，メモリには入れる必要がない。右上のＲおよびＳの部分は前進段階においては処理済みの部分で，後退代入の段階までは処理に関係ないので，別のテーブルにする。ＡおよびＢが前進段階においてこれから処理される部分である。前進の計算式は，a_{pp} をピボットとするとき，

$$a'_{ij}=a_{ij}-a_{ip}\cdot(a_{pj}/a_{pp})$$

であるが，右辺に零がある場合の省略計算式を考えてみると，

$$\left.\begin{array}{ll} a_{ij}=0 & \text{ならば}\quad a'_{ij}=-a_{ip}\cdot(a_{pj}/a_{pp}) \\ a_{ip}=0 & \text{ならば}\quad a'_{ij}=a_{ij} \\ a_{pj}=0 & \text{ならば}\quad a'_{ij}=a_{ij} \end{array}\right\}\ \text{(不要)}$$

となるので，この判定に従って最小限の計算を行なう。また，この判定を簡単にするため，あらかじめピボット列およびピボット行の非零要素だけを集めて，AIP および APJ というノンゼロテーブルを作っておく。右辺についても同様にする。前進段階においては，ＲおよびＳの部分は新しく追加されゆくだけであるが，ＡおよびＢの部分は全面的に変形を受け，新しい非零要素が誕生するたびにノンゼロテーブル全体を書きかえなければならない（順序を無視すれば新しい要素を末尾に追加するだけでよいのであるが，掃出法の計算をスムースに行なうためには行，列の順に正しく並んでいないとぐあいが悪い）。このため，ＡのほかにＡ1というメモリ・スペースをとっておいて，ノンゼロテーブルＡから要素をとり出して計算した結果を Ａ1に入れる。そして１段階終るごとに Ａ1 のデータをＡに移して（または交互に名前をつけ変えて）計算を続ける。右辺に関しても同様の処置を行なう。

前進段階が終了するとＡとＢの部分は空になり，すべてのデータはＲとＳに入っていることになる。後退代入の段階では一番下の行から始めて，Ｒの要素を１個ずつ消去してゆくが，その際にＲの中の他の要素は変形を受けず，右辺のＳの要素が変形を受ける。この時にまた非零要素が誕生することになるので，変形後の値を別の領域Ｑに記録する。こうしてＲの要素をすべて消去すればＳの部分に解Ｘが得られるわけである。

基本原理は以上のとおりであるが，実際には，演算中に起る各種のスペシャルケースの対策が必要で，このために実際のプログラムは一段と複雑になっている。たとえば APJ あるいは AIP のテーブルが空の場合の対策などが必要であり，その他，計算不能のチェックなども必要である。

ノンゼロテーブルを使用する以上は，非零要素の個数が全要素個数に比較して十分に小さいものでなければ意味がない。反復法の場合には，マトリクスの要素は計算を始めてから終りまで変化せず，解の近似ベクトルが変化するだけなので問題ないが，掃出法の場合には計算が進むに従って，零でない要素の個数がしだいに増加する。たとえば（極端な例であるが）

$$\left\{\begin{array}{rcl} 2x_1+2x_2+2x_3+2x_4 &=& 2 \\ 2x_1+3x_2 &=& 0 \\ 2x_1+3x_3 &=& 0 \\ 2x_1+3x_4 &=& 0 \end{array}\right.$$

の場合，機械的に掃出法の公式で処理すると，第１段階ですでに（第２，３，４行から第１行を引いて）

$$\left\{\begin{array}{rcl} 2x_1+2x_2+2x_3+2x_4 &=& 2 \\ x_2-2x_3-2x_4 &=& -2 \\ -2x_2-x_3-2x_4 &=& -2 \\ -2x_2-2x_3-x_4 &=& -2 \end{array}\right.$$

となり，新たに処理すべき部分は零でない要素ばかりになってしまう。このようなことが起らないためには，たとえばマトリクスが図４のような形であればよい。ただし斜線の部分は零でない要素，白地の部分は零。この形が block diagonal と呼ばれているものである。必ずしもこの形でなくとも，たとえば行または列の

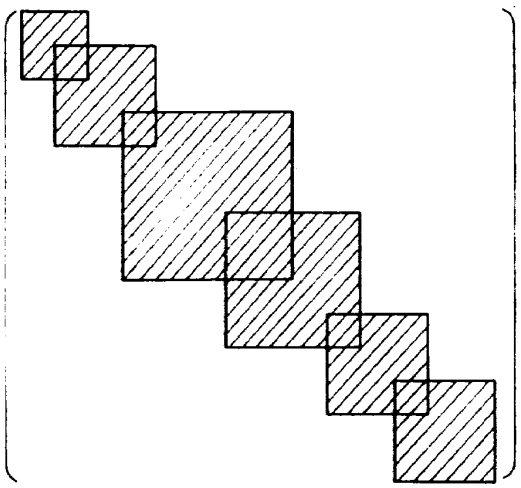

図 4　block diagonal の例

入れかえによって図4の形になるもの，あるいは下三角行列に近い形のもの，などに対しても LEAZ は有効である。

LEAZ

目的

AはN行N列のマトリクス，BとXはN行M列のマトリクスとするとき，方程式

$$AX=B$$

をXについて解く。すなわち$X=A^{-1}B$を求める。

解法

掃出法，ノンゼロテーブル使用

適用条件

次数その他の制限は特にないが，9×M×Nがコア容量を越えない事が必要である。なお，零要素の割合が多いマトリクスでないと不利になる。

呼び方

LEAZ (AIJT, IAIJT, JAIJT, BIJT, IBIJT, JBIJT, KAZET, KBZET, LA, LR, N, M, MM, NN, MN, APJT, JAPJT, BPJT, JBPJT, IAIPT, AIPT, RIJT, IRIJT, JRIJT, SIJT, ISIJT, JSIJT, KRZET, AIJT1, IAIJT1, JAIJT1, BIJT1, IBIJT1, JBIJT1, KAZET1, KBZET1, KSZET, Q)

使用サブルーチン

なし

引数

引数名	実行前	実行後	次元	型
AIJT	係数行列A (注*)	(破壊される)	LA	実
IAIJT	同，行番号	(破壊される)	LA	整
JAIJT	同，列番号	(破壊される)	LA	整
BIJT	右辺行列B (注*)	解Xが入る	MN	実
IBIJT	同，行番号	同，行番号	MN	整
JBIJT	同，列番号	同，列番号	MN	整
KAZET	Aの列区切 (注**)	(破壊される)	NN	整
KBZET	Bの列区切 (注**)	(破壊される)	NN	整
LA	AIJTのテーブルサイズ (注†)	(不変)	—	整
LR	RIJTのテーブルサイズ (注††)	(不変)	—	整
N	Aの行の数	(不変)	—	整
M	Bの列の数	(不変)	—	整
MM	M＋1	(不変)	—	整
NN	N＋1	(不変)	—	整
MN	M×N	(不変)	—	整
APJT	(関係なし) (注†††)	(破壊される)	N	実
JAPJT	〃	〃	N	整
BPJT	〃	〃	M	実
JBPJT	〃	〃	M	整
IAIPT	〃	〃	N	整
AIPT	〃	〃	N	実
RIJT	〃	〃	LR	実
IRIJT	〃	〃	LR	整
JRIJT	〃	〃	LR	整
SIJT	〃	〃	MN	実
ISIJT	〃	〃	MN	整
JSIJT	〃	〃	MN	整
KRZET	〃	〃	NN	整
AIJT1	〃	〃	LA	実
IAIJT1	〃	〃	LA	整
JAIJT1	〃	〃	LA	整
BIJT1	〃	〃	MN	実
IBIJT1	〃	〃	MN	整
JBIJT1	〃	〃	NM	整
KAZET1	〃	〃	NN	整
KBZET1	〃	〃	MM	整
KSZET	〃	〃	NN	整
Q	〃	〃	M	実

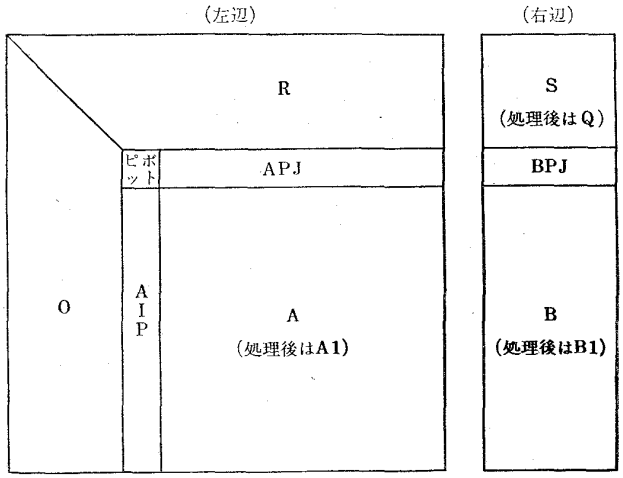

図 5 掃出法におけるノンゼロテーブル

* ノンゼロテーブル表現による（1.2参照）。テーブル内の配列は列番号の小さいものから順に，また同じ列番号のものの中では行番号の順でなければいけない。

** 上記の配列によれば，最初に第1列の非零要素がいくつか並び，次に第2列，続いて第3列……という順になるが，その第j列の最初の要素の，ノンゼロテーブル内の添字を KAZET(l)（Bに関しては KBZET(j)）に入れておく。言いかえれば，第$j-1$列までに含まれる非零要素の個数プラス1が KAZET(j)，KBZET(j)である。

† AIJT には，実行前には係数行列が入っているが，計算中は図5のAの部分が入る。したがって，ここに収容される非零要素の個数は，問題によっては，一時的にかなり増大する可能性がある。LAとしては，この増大した時に予想される最大個数を書く。たとえば図4のような block diagonal の形ならば図の斜線部分以外に非零要素が出る可能性はないから，斜線部分の要素数（たとえ最初に零であってもそれを含めて数えて）をLAとしておけば安全である。

†† RIJT は，図5のRの部分の非零要素を収容するノンゼロテーブルである。ここに収容される要素の個数は，前進段階では1ステップごとに増加していって，前進段階の終りで最大になる。その時に予想される要素の個数をLRとして書くのであるが，LAのの半分ぐらいとみておけばよいであろう。

††† いずれも計算の中間結果を収容するワーキングスペースであるが，adjustable dimension にするために引数に入れてある。したがってメインプログラムで dimension を切っておかなければならない。ワーキングスペースの変数名の意味は，

APJT, JAPJT	図5の APJ
BPJT, JBPJT	図5の BPJ
AIPT, IAIPT	図5の AIP
RIJT, IRIJT, JRIJT	図5のRの部分
SIJT, ISIJT, JSIJT	図5のSの部分
KRZET	Rの行区切番号
AIJT1, IAIJT1, JAIJT1	図5のA1
BIJT1, IBIJT1, JBIJT1	図5のB1
KAZET1	A1の列区切番号
KBZET1	B1の列区切番号
KSZET	Sの行区切番号
Q	図5のQの部分

いずれもノンゼロテーブルとその指標である。

〔III〕 ユーティリティ・ルーチン

ここでは，連立一次方程式や逆行列計算の，計算前の処理，または計算後の処理に用いるために作成した補助的なサブルーチンについて述べる。

個々の計算の内容は，

ベクトルの内積
ベクトルにマトリクスを掛ける
マトリクスの積
小行列から大きなマトリクスを組み立てる
ノンゼロテーブルを作る

などの，比較的簡単なものであるから，そのつど自分でコーディングすることもできるはずであるが，マトリクスが磁気ドラムや磁気テープに入っている場合などはかなりめんどうであるし，サブルーチン形式にしておく方がプログラムが明快になりディバッグが簡単になる。また，ここに収録したサブルーチンの多くは，その最重要部分を HISAP で書いているので時間的にも有利である。

2.9 コアメモリ関係

最初の三つ，

VECMUL, MAMULT, TMAVEM

は，マトリクスとベクトルの演算で，HISAP で書いてあるので速い，ということが特徴である。

残りの二つ，

MADIA, NORMA

は，連立方程式または逆行列計算の準備をするためのプログラムで，MADIA は GSP で連立一次方程式を解く前に対角要素を1にするためのもの，NORMA は掃出法の計算の前に要素の絶対値を平均化してオーバーフローを防止するためのものである。

VECMUL

目的

ベクトルの内積を求める

$$S=(x\cdot y)=\sum_{i=1}^{n}\xi_i\eta_i$$

呼び方

CALL VECMUL (X, Y, N, S)

引数

引数名	実行前	実行後	次元	型
X	ベクトル x	(不変)	N	実
Y	ベクトル y	(不変)	N	実
N	次元	(不変)	—	整
S	(関係なし)	内積 s	—	実

備考

同じサブルーチン名で，ゼロジャンプ法を用いたプログラムを作成してある。零が多い場合には，これを用いると時間が短縮される。この場合，零要素の多い方をXにすること。

MAMULT

目的

ベクトルにマトリクスを掛ける

$$Ax=y$$

呼び方

CALL MAMULT (A, X, Y, N)

引数

引数名	実行前	実行後	次元	型
A	マトリクスA	(不変)	N×N	実
X	ベクトル x	(不変)	N	実
Y	(無関係)	結果 y	N	実
N	次元	(不要)	—	整

備考

同じサブルーチン名で，ゼロジャンプ法を用いたプログラムを作成してある。

TMAVEM

目的

ベクトルに転置行列を掛ける

$$A^T x=y$$

呼び方

CALL TMAVEM (A, X, Y, N)

引数，備考

MAMULT の場合と同じ

MADIA

目的

連立一次方程式

$$Ax=C$$

の両辺を（行ごとに）常数（a_{ii}）で割って，係数行列の対角要素を1にする。

呼び方

CALL MADIA (A, C, N)

引数

引数名	実行前	実行後	次元	型
A	係数行列A	結果A′	N×N	実
C	右辺C	結果C′	N	実
N	次数	不変	—	整

NORMA

目的

連立一次方程式

$$Ax=C$$

に下記の操作をほどこして変換する。

まず，両辺を，行ごとに，その行のAの要素の絶対値最大のものの絶対値で割る。すなわち，その結果を $A'=(a_{ij})$ および $C'=(C_i)$ で表わせば

$$a'_{ij}=\frac{a_{ij}}{\max_j|a_{ij}|},\qquad c'_i=\frac{c_i}{\max_j|a_{ij}|}$$

次に上で得られた A' を，列ごとに，その列の要素の絶対値最大のものの絶対値で割る。すなわち，その結果を $A''=(a''_{ij})$ で表わせば，

$$a''_{ij}=\frac{a'_{ij}}{\max_i|a'_{ij}|}$$

この変換により，係数行列のすべての要素は，絶対値1以下になり，かつ，どの行にも，どの列にも絶対値が1になる要素が少なくとも1個あるようになる。ただし，方程式

$$Ax=C \qquad \cdots\cdots(1)$$

の解と，

$$A'x=C' \qquad \cdots\cdots(2)$$

の解は同一であるが（右辺を常数で割っただけである故），第2の変換後の方程式

$$A''x=C' \qquad \cdots\cdots(3)$$

の解は，もとの方程式の解と異なるので，これで解いた場合にはそのあとで補正の計算を行なう必要がある。補正の方法は，(1)の解を

$$x_1=(\xi_1,\ \xi_2,\ \cdots\cdots\xi_n)$$

(3)の解を

PDRUM

目的

ドラムに格納されているマトリクスをプリントする

適用条件

サイズ 200×200 まで

呼び方

CALL PDRUM (N, INDEX)

引数*

Nは次数（5＜N）

INDEX はマトリクスの指定で，

マトリクスAをプリントしたいときは1

マトリクスBをプリントしたいときは2

マトリクスCをプリントしたいときは3

とする。ただし，A, B, C

DRUM COMMON A(40000), B(40000), C(40000)

で受け渡す。

使用サブルーチン

なし

プリントフォーム

row	col. 1	·····	col. 5
1	M (1, 1)	·····	M (1, 5)
2	M (2, 1)	·····	M (2, 5)
3	M (3, 1)	·····	M (3, 5)
	·····		·····
	·····		·····
N	M (N, 1)	·····	M (N, 5)
row	col. 6	·····	col. 10
1	M (1, 6)	·····	M (1, 10)
2	M (2, 6)	·····	M (2, 10)
3	M (3, 6)	·····	M (3, 10)
	·····		·····
	·····		·····

(以下同様)

MADIAD

目的

連立一次方程式

$$Ax = C$$

の両辺を（行ごとに）常数（a_{ii}）で割って，係数行列の対角要素を1にする。

適用条件

200×200 まで

呼び方

CALL MADIAD (C, N, D, W)

引数*

引数名	実行前	実行後	次元	型
C	右辺	変換された右辺	N	実
N	次数	(不変)	—	整
D	対角要素	(不変)	N	実
W	(関係なし)	(破壊される)	N	実

NORMAD

目的

連立一次方程式

$$Ax = C$$

の係数行列と右辺をノーマライズする。詳細は19ページの NORMA 参照。

適用条件

200×200 まで

呼び方

CALL NORMAD (C, F, N)

引数*

引数名	実行前	実行後	次元	型
D	右辺	変換された右辺	N	実
F	(関係なし)	補正用係数	N	実
N	次数	(不変)	—	整

2.11 磁気テープ関係

以下に述べる4個のサブルーチンは，磁気テープを用いて連立方程式を解くプログラム(2.4) のためのデータを準備するためのものである。また，計算の際には磁気テープを使用せずノンゼロテーブルによるとしても，ノンゼロテーブルにする以前の普通の形のマトリクスとしてデータを準備するためにも，本節のサブルーチンが有効である。すなわちこの場合には，

マトリクスを組み立てる (2.11)

ノンゼロテーブルに変換する (2.12)

方程式を解く (2.2)

の順に使用される。

これらのルーチンにおける磁気テープの記録様式は共通で，順序は columnwise, 1列分1レコード(1列分は必ず1個の WRITE 命令で書き込む) という形式にしてある。

〔注意事項〕

磁気テープを使用する場合の一般的注意として，下

* 20ページの注意事項参照

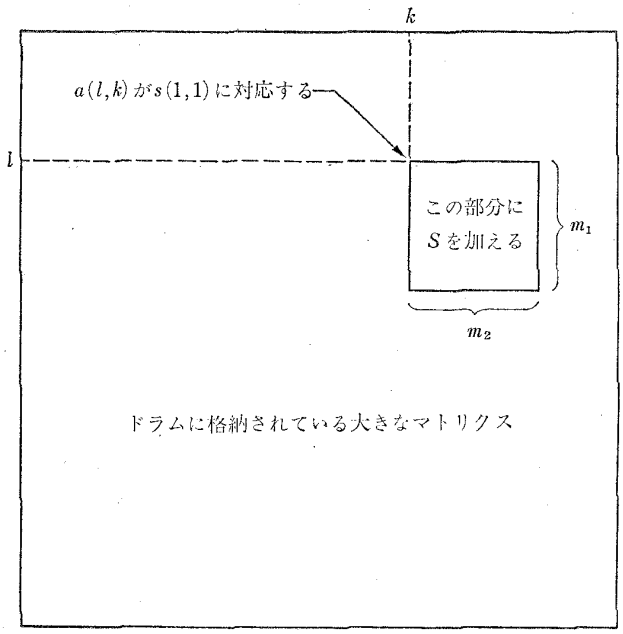

図 6　ASMTD の引数の意味

引数*

引数名	実　行　前	実　行　後	次元	型
S	小行列S	(不変)	M1×M2	実
M1	Sの行数	(不変)	—	整
M2	Sの列数	(不変)	—	整
K	(図6参照)	(不変)	—	整
L	(〃)	(不変)	—	整
N	Aの次数	(不変)	—	整

使用サブルーチン

なし

MDMTV

目的

ベクトルにマトリクスを掛ける

$$Ax = y$$

適用条件

Aのサイズ 200×200 まで

呼び方

CALL MDMTV (X, Y, N, 200)

引数*

引数名	実　行　前	実　行　後	次元	型
X	ベクトル x	(不変)	N	実
Y	(無関係)	結果 y	N	実
N	Aの次数	(不変)	—	整

使用サブルーチン

なし

* 20ページの注意事項参照

備考

同じサブルチーン名で，ゼロジャンプ法を用いたプログラムとゼロジャンプ法を用いないプログラムとを用意してある。使用法は全く同じ。

MDMTVT

目的

ベクトルに転置マトリクスを掛ける。

$$A^T x = y$$

適用条件

Aのサイズ 200×200 まで

呼び方

CALL MDMTVT (X, Y, N, 200)

引数*

引数名	実　行　前	実　行　後	次元	型
X	ベクトル x	(不変)	N	実
Y	(無関係)	結果 y	N	実
N	Aの次数	(不変)	—	整

使用サブルーチン

VECMUL

備考

この MDMTVT の方が前記の MDMTV より計算時間が短かい。したがって対称行列の乗算には MDMTVT を用いると有利である。

MMDMT

目的

ドラムに格納されている二つのマトリクスの積を作る。A・B＝C

適用条件

200×200 まで

呼び方

CALL MMDMT (N, Y)

引数*

引数名	実　行　前	実　行　後	次元	型
N	次数	(不変)	—	整
Y	(無関係)	(破壊される)	N	実

マトリクスは

DRUM COMMON A(40000), B(40000), C(40000)

で受け渡す。内部表現等については20ページの注意事項参照。

使用サブルーチン

MDMTV

PDRUM

目的

ドラムに格納されているマトリクスをプリントする

適用条件

サイズ 200×200 まで

呼び方

CALL PDRUM (N, INDEX)

引数*

Nは次数（5＜N）

INDEX はマトリクスの指定で，

マトリクスAをプリントしたいときは1

マトリクスBをプリントしたいときは2

マトリクスCをプリントしたいときは3

とする。ただし，A, B, C

DRUM COMMON A(40000), B(40000), C(40000)

で受け渡す。

使用サブルーチン

なし

プリントフォーム

row	col. 1	·····	col. 5
1	M (1, 1)	·····	M (1, 5)
2	M (2, 1)	·····	M (2, 5)
3	M (3, 1)	·····	M (3, 5)
	·····		·····
	·····		·····
N	M (N, 1)	·····	M (N, 5)
row	col. 6	·····	col. 10
1	M (1, 6)	·····	M (1, 10)
2	M (2, 6)	·····	M (2, 10)
3	M (3, 6)	·····	M (3, 10)
	·····		·····
	·····		·····

(以下同様)

MADIAD

目的

連立一次方程式

$$Ax = C$$

の両辺を（行ごとに）常数 (a_{ii}) で割って，係数行列の対角要素を1にする。

適用条件

200×200 まで

呼び方

CALL MADIAD (C, N, D, W)

引数*

引数名	実行前	実行後	次元	型
C	右辺	変換された右辺	N	実
N	次数	(不変)	—	整
D	対角要素	(不変)	N	実
W	(関係なし)	(破壊される)	N	実

NORMAD

目的

連立一次方程式

$$Ax = C$$

の係数行列と右辺をノーマライズする。詳細は19ページの NORMA 参照。

適用条件

200×200 まで

呼び方

CALL NORMAD (C, F, N)

引数*

引数名	実行前	実行後	次元	型
D	右辺	変換された右辺	N	実
F	(関係なし)	補正用係数	N	実
N	次数	(不変)	—	整

2.11 磁気テープ関係

以下に述べる4個のサブルーチンは，磁気テープを用いて連立方程式を解くプログラム(2.4) のためのデータを準備するためのものである。また，計算の際には磁気テープを使用せずノンゼロテーブルによるとしても，ノンゼロテーブルにする以前の普通の形のマトリクスとしてデータを準備するためにも，本節のサブルーチンが有効である。すなわちこの場合には，

マトリクスを組み立てる (2.11)

ノンゼロテーブルに変換する (2.12)

方程式を解く (2.2)

の順に使用される。

これらのルーチンにおける磁気テープの記録様式は共通で，順序は columnwise, 1列分1レコード(1列分は必ず1個の WRITE 命令で書き込む) という形式にしてある。

〔注意事項〕

磁気テープを使用する場合の一般的注意として，下

* 20ページの注意事項参照

記の手続きを忘れないようにすること。

(1) ＩＤカードで，使用するユニットの番号を指定しておくこと。たとえば

$ 登録番号　ＩＤ　HIJOB, /2/

(2) ＩＤカードの次に EQUAL カードを入れてユニット番号（プログラム上の）と装置番号（ハードウェアの番号）の対応を指定する。たとえば

$ SYSUT 2　EQUAL　E 1

(3) オペレーション指示カード（デックの一番上にのせる紫色の線の入ったカード）に，磁気テープを使用する旨，赤字で大きく書いておくこと。

(4) ユニット番号は１，２，３，４，８，９の内のいずれか，装置番号はE1, E2, E4, E4, E5 の内のいずれかを選ぶことができるが，同時処理される他のプログラム（たとえばＸＹプロッタの制御ルーチン）との関係があるので，あらかじめオペレータと打合わせをしてから決めることが望ましい。（そうしないと，特別な時間にしか計算機にかけられないことになる。）

(5) ユニット番号と EQUAL カード左辺の対応は下記のとおりである。

SYSUT1	はユニット番号	1
SYSUT2	〃	2
SYSUT3	〃	3
SYSUT4	〃	4
SYSUT5	〃	8
SYSUT6	〃	9

詳細について疑問の点はプログラム相談室に照会されたい。資料としては HITAC 5020, 5020 E/F オペレーティング・システム　プログラム・マニュアル (昭42.4)　参照

CLEART

目的

磁気テープにＮ×Ｎの零マトリクスを入れる

呼び方

CALL　CLEART (N, W, K)

引数

引数名	実　行　前	実　行　後	次元	型
N	次数	(不変)	—	整
W	(関係なし)	(破壊される)	N	実
K	ユニット番号	(不変)	—	整

ASMTT

目的

磁気テープに格納されている大きなマトリクス $A=(a_{ij})$ の一部分に，コアメモリの中の小さなマトリクス $S=(s_{ij})$ を加える(図６)。結果を $A'=(a'_{ij})$ で表わすと

$$a'_{ij}=\begin{cases} a_{ij}+s_{(i-l+1),(j-k+1)} & \begin{pmatrix} l\leqq i<l+m_1 \\ k\leqq j<k+m_2 \end{pmatrix} \\ a_{ij} & \text{(その他の部分)} \end{cases}$$

適用条件

Ｓのサイズ　30×30 まで

呼び方

CALL　ASMTT (S, M1, M2, K, L, N, C, K1, K2)

引数

引数名	実　行　前	実　行　後	次元	型
S	小行列Ｓ	(不変)	M1×M2	実
M1	Ｓの行数	(〃)	—	整
M2	Ｓの列数	(〃)	—	整
K	列位置(図6)	(〃)	—	整
L	行位置(図6)	(〃)	—	整
N	Ａの次数	(〃)	—	整
K1	Ａ′のユニット番号	(〃)	—	整
K2	Ａのユニット番号	(〃)	—	整
C	(関係なし)	(破壊される)	N	実

備考

磁気テープから，その一部分のデータを読み出して計算した後，同じテープの同じ場所にすぐ書き込むことは（その以後のデータを破壊する危険があるため）文法上許されていない。このため，Ａが格納されているテープからデータを読んで計算した結果（Ａ′）は，別のテープに書き込まなければならない。このために二つのユニット番号が必要になるのである。K2 の方からデータを読み，Ｓを加え，結果は K1 に書き込む。

MTMTV

目的

磁気テープに格納されているマトリクスを，コアメモリの中のベクトルに掛ける

$$Ax=y$$

呼び方

CALL　MTMTV (M, N, X, Y, K, W)

引数

引数名	実行前	実行後	次元	型
M	Aの行数	(不変)	—	整
N	Aの列数	(〃)	—	整
X	ベクトルX	(〃)	N	実
Y	(関係なし)	結果Y	M	実
K	ユニット番号	(不変)	—	整
W	(関係なし)	(破壊される)	N	実

備考

ゼロジャンプをする方式と，ゼロジャンプをしない方式の二種類のデックを作ってある，

MTMTVT

目的

磁気テープに格納されているマトリクスAの転置行列を，コアメモリの中のベクトルxに掛ける

$$A^T x = y$$

呼び方

CALL MTMTVT (M, N, X, Y, K, W)

引数，備考

MTMTV に同じ

2.12 ノンゼロテーブル関係

TNZMVM

目的

ノンゼロテーブルの形で表現されているマトリクスAの転置行列をベクトル x に掛ける。

$$A^T x = y$$

呼び方

CALL TNZMVM (AIJ, IAIJ, JAIJ, K, M, N, X, Y)

引数

引数名	実行前	実行後	次元	型
AIJ	非零要素	(不変)	K	実
IAIJ	その行番号	(〃)	K	整
JAIJ	同，列番号	(〃)	K	整
K	非零要素個数	(〃)	—	整
M	Aの行数	(〃)	—	整
N	Aの列数	(〃)	—	整
X	ベクトル x	(〃)	M	実
Y	(無関係)	結果 y	N	実

NZMVM

目的

ノンゼロテーブルの形で表現されているマトリクス Aをベクトルに x 掛ける

$$Ax = y$$

呼び方

CALL NZMVM (AIJ, IAJ, JAIJ, K, M, N, X, Y,)

引数

TNZMVM に同じ

NZEMDC

目的

ドラムに格納されているマトリクスAを，ノンゼロテーブルの形に変換してコアに入れる

呼び方

CALL NZEMDC (M, N, AIJ, IAIJ, JAIJ, K, W)

引数 (20ページの注意事項参照)

引数名	実行前	実行後	次元	型
M	Aの行数	(不変)	—	整
N	Aの列数	(〃)	—	整
AIJ	(関係なし)	Aの非零要素	K	実
IAIJ	(〃)	その行番号	K	整
JAIJ	(〃)	その列番号	K	整
K	予想される非零要素個数の上限	非零要素個数	—	整
W	(関係なし)	(破壊される)	M	実

備考

AIJ 等の次元Kは実行前のK

NZEMTC

目的

磁気テープに格納されているマトリクスAを，ノンゼロテーブルの形に変換してコアに入れる

呼び方

CALL NZEMTC (M,N, AIJ,IAIJ,JAIJ, K,W, L)

引数

引数名	実行前	実行後	次元	型
M	Aの行数	(不変)	—	整
N	Aの列数	(不変)	—	整
AIJ	(関係なし)	Aの非零要素	実行前のK	実
IAIJ	(〃)	その行番号	〃	整
JAIJ	(〃)	同，列番号	〃	整
K	非零元要素個数上限の予想値	非零要素個数	—	整
W	(関係なし)	(破壊される)	M	実
L	ユニット番号	(不変)	—	整

第3部　計算例と所要時間

3.1　計算例に用いた例題について

サブルーチンのテストや数値解法の比較のためによく用いられる「標準的問題」というものがある。これらの標準的問題は厳密解がわかっており，特別にむつかしすぎるとか易しすぎるとかいうような不自然さがなく，問題のデータ（たとえば連立方程式の係数）の作成が機械的に簡単にできる，などの利点があるので便利である。

ここでは，標準的問題の中から，つぎの五つの問題を選んで用いる。

<問題1>連立一次方程式

$$\begin{pmatrix} 1 & \alpha & & & 0 \\ \alpha & 1 & \alpha & & \\ & \alpha & 1 & \alpha & \\ & & \ddots & \ddots & \ddots \\ 0 & & & \alpha & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} 1+\alpha \\ 1+2\alpha \\ 1+2\alpha \\ \cdots\cdots \\ 1+\alpha \end{pmatrix}$$

正解は　$x_1 = x_2 = \cdots\cdots x_n = 1$

計算例では $\alpha=0.2$ または $\alpha=0.4$ とした。その場合には係数行列は正値となる。また明らかに対称であるから，正値対称行列の場合の解法のテストに使用できる。

なお，α の値が大きいほど解きにくくなる。すなわち，反復法の収束が悪く，掃出法の場合の有効数字の桁落ちも起りやすくなることが知られている。

<問題2>連立一次方程式

$$\begin{pmatrix} 1 & 0.2 & 0.01 & 0.01 & & 0 \\ 0.2 & 1 & 0.2 & 0.01 & 0.01 & \\ 0.01 & 0.2 & 1 & 0.2 & 0.01 & \ddots \\ 0.01 & 0.01 & 0.2 & 1 & 0.2 & \ddots \\ & 0.01 & 0.01 & 0.2 & 1 & \ddots \\ 0 & & \ddots & \ddots & \ddots & \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} 1.22 \\ 1.42 \\ 1.43 \\ 1.44 \\ 1.44 \\ \vdots \end{pmatrix}$$

正解は　$x_1=x_2=\cdots\cdots=x_n=1$

問題1の $\alpha=0.2$の場合よりは，やや解きにくい。

この係数行列も正値対称である。

<問題3>連立一次方程式で，係数行列が非対称な場合の例で，係数行列は

$$\begin{pmatrix} 1 & 0.25 & & & & a_{1n}\text{のみ}1\text{他は}0 & 1 \\ 0.25 & 1 & 0.25 & & & & \\ & 0.25 & 1 & 0.25 & & & \\ & & 0.25 & 1 & 0.25 & & \\ & & & \ddots & \ddots & \ddots & \\ 0 & & & & 0.25 & 1 & \end{pmatrix}$$

右辺は正解が $x_1=x_2=\cdots\cdots=x_n=1$ になるように定める。

<問題4>問題2と同様，ただし非対称にするために係数行列を一部変更して

$$a_{1n}=0.01,\quad a_{n1}=0$$

としたもの。

<問題5>逆行列を求める。もとのマトリクスは

$$\begin{pmatrix} n & n-1 & n-2 & n-3 & & 1 \\ n-1 & n-1 & n-2 & n-3 & & 1 \\ n-2 & n-2 & n-2 & n-3 & & 1 \\ n-3 & n-3 & n-3 & n-3 & & 1 \\ & & \cdots\cdots & & & \\ 1 & 1 & 1 & 1 & \cdots & 1 \end{pmatrix}$$

これの逆行列（正解）は

$$\begin{pmatrix} 1 & -1 & & & & 0 \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 2 & -1 \\ 0 & & & & -1 & 2 \end{pmatrix}$$

3.2　50元の連立一次方程式を解いた例

問題1で $\alpha=0.4$とし，反復法で6桁の精度が得られるまで計算を行なった。第2部の各種のサブルーチンを用いた場合の所要時間は，つぎのとおりである。ただし所要時間は CALL CLOCK により測定したもので，問題作成時間，プリント時間を含まない。

使用したサブルーチン名	ゼロジャンプ法のデックを用いた場合	ゼロジャンプ法のデックを用いない場合
CGSP	2秒	5秒
GSP	4秒	7秒
SDSP	6秒	15秒

なお参考までに日立の技術計算ライブラリ[1]のサブル

ーチンを使用した場合の結果を示すと，次のとおりである。

LINGS 3秒
LINCG 77秒

最後の LINCG に非常に時間がかかっているのは，これが正値対称行列専用でなく，一般用の公式を用いているためである。

つぎに，非対称の問題の例として，問題3を，第二部の CGN と技術計算ライブラリの LINCG で解いたときの所要時間をあげる。

CGN 6秒
LINCG 46秒

同じ LINCG でも先の場合と時間が異なるのは，問題の相違によるものである。

精度と所要時間の関係については次項 (3.3) 参照。

3.3 200 元の連立一次方程式を解いた例（磁気ドラム使用）

問題1，$\alpha=0.4$，精度6桁とした場合，

CGSPD 54秒

であった。反復1回当りの所要時間は約3秒で，先の50元の場合と同じ時間（約0.1秒）と比較すると，磁気ドラムを使用したためにかなり遅くなっている。計算量は50元と 200 元で4倍の差があるが，50元の場合の0.1秒を4倍しても 0.4 秒にしかならないから200元の場合に3秒というのは主として磁気ドラムの待時間によるものであることがわかる。

精度と反復回数の関係は，解法と問題によって差があるが，問題1で $\alpha=0.2$ とした場合の各種解法の収束の状況はつぎのとおりである。（いずれも 200 元。10^{-3} 等の数字は，相対誤差の概数）

反復回数	SD法	GS法	PJ法	CG法*
3回	10^{-3}	10^{-2}	10^{-1}	10^{-3}
6回	10^{-4}	10^{-4}	10^{-2}	10^{-4}
9回	10^{-5}	10^{-6}	10^{-4}	10^{-5}

ここでPJ法というのは point Jacobi と呼ばれている解法で，収束が悪いので第2部には収録しなかった。なお，収束の速さは上に述べた諸条件に非常に大きく左右されるので，上の例は一つの参考にすぎない。

3.4 500 元の連立一次方程式を解いた例（磁気テープ使用）

問題2，精度5桁の場合

CGSPT 315秒

また非対称の例としては問題4，精度5桁の場合

CGNT 1,322 秒

であった。

* CG法のみは $\alpha=0.4$の場合のデータである。問題の条件が少し異なる点に注意。

3.5 ノンゼロテーブル法の効果

ノンゼロテーブル法を使用すると計算量が少なくなるばかりでなく，通常の表現ではコアメモリに収容できないような大きなマトリクスをコアメモリの中に入れて計算できるため，磁気ドラムや磁気テープの待時間がなくなり，時間的に非常に有利となる。その実例を二三あげると，まず，問題1で $\alpha=0.4$，200元，精度6桁としてノンゼロテーブル法 NZCGSP と普通の方法 CGSPD (200元なので磁気ドラム使用) を比較すると，

NZCGSP 5秒
CGSPD 54秒

で約10倍の差がある。また同じく200元で，問題3 (非対称の例) を，ノンゼロテーブル法の NZCGN で解いた場合と普通の方法 CGND で解いた場合についてみると，

NZCGN 20秒
CGND 416秒

で約20倍の差がある。つぎに磁気テープを使用するサブルーチンとの比較例として，問題2，500元，精度5桁の場合，ノンゼロテーブル法の NZCGSP と，磁気テープ使用の普通の方法 CGSPT とでは，

NZCGSP 4秒
CGSPT 315秒

で実に80倍近い差が見られる。

以上の例にみられるとおり，ノンゼロテーブル法は非常に有利であるから，200元以上の方程式でも非零要素の個数がコアメモリに収容できる程度ならば必ずノンゼロテーブル法を使することを推奨する。

3.6 逆行列計算の例

問題5を使用し，サイズ50×50の場合，所要時間は

MAINV 1 17秒
MAINV 2 27秒

であった。精密計算用の MAINV 3 の所要時間は要求精度により異なるが，精度改善のためのくりかえし計算1回につき約50秒（50×50の場合）かかるので，その数倍程度と考えておけばよい。

MAINV 3 による精度改善の効果については，問題でサイズ100×100の場合，

改善前 精度5桁
改善3回 6桁

改善8回　7桁

という結果が出ている。

3.7 ユーティリティ・ルーチンの所要時間

ユーティリティ・ルーチンのおもなものの所要時間はつぎのとおりである（いずれも実測値）。

VECMUL（ベクトルの内積）

N＝1000の場合

零要素が多い例では　28ミリセカンド

零要素が全くない場合　76ミリセカンド

MAMULT（マトリクスとベクトルの積）

N＝100の場合

零要素が多い例では　370ミリセカンド

零要素が全くない例　910ミリセカンド

MDMTV（磁気ドラムに格納されているマトリクスとベクトルの積）

N＝50の場合

零要素が全くない例　640ミリセカンド

なお

VECMUL の所要時間は N に比例する

MAMULT の所要時間は N^2 に比例する

掃出法による計算の時間は N^3 に比例する

から，これによって他のサイズの場合の所要時間を推定計算することができる。

文　献

1) HITAC 5020 技術計算用ライブラリー，(昭和40) 日立製作所

2) FRAN 骨組立体構造物の解析プログラム，(昭和42) 日本アイビーエム

3) 吉峯鼎，伊東浩，荒井汎；車体側構強度計算の電子計算機による高速化，日本機械学会論文集32巻239号 (昭和41)

4) 清水辰次郎；実用数学，(昭和36) 朝倉書店

5) R. A. Frazer, W. J. Duncan and A. R. Collar; Elementary matrices and some applications to dynamics and differential equations, (1955), Cambridge University Press.

6) E. Isaacson and H. B. Keller ; Analysis of numerical methods, (1966), John Wiley and Sons

7) E. Kosko ; Matrix inversion by partitioning, Aeronautical Quarterly, vol. 9, No. 2, (1957)

8) H. I. Meyer and B. J. Hollingsworth; A method of inverting large matrices of special form, Mathematical Tables and other Aids to Computation, vol. 11, no. 58, (1957)

9) V. Bargmann, D. Montogomery and J. von Neumann ; Solution of linear systems of high order, Report prepared for the Bureau of Ordnance (NORD-9596)

10) J. von Neumann and Goldstine ; Numerical inverting of marices of high order, Bull. Amer. Math. Soc., vol. 53, (1947)

TM-106 動安定微係数測定用風洞天秤について	1967年5月	高島一明，榊原盛三 原亘利，北出大三 金成正好
TM-107 プロペラ後流偏向型STOL機の風洞試験（II）	1967年6月	犬丸矩夫，岡部祐二郎 北村清美，川幡長勝 高橋侔，木村友昭
TM-108 インダクタンス型小型圧力変換器の試作とその応用	1967年7月	長洲秀夫，柳沢三憲
TM-109 ロケットの飛しょう径路計算のためのプログラム	1967年7月	戸川隼人，石黒登美子
TM-110 二次元スラットおよびスロッテッドフラップの実験的研究（I）	1967年8月	犬丸矩夫，北村清美 川幡長勝
TM-111 リフトジェットVTOL機の離陸径路に関する近似解	1967年8月	西村博史
TM-112 極超音速風洞ペブル加熱器の特性	1967年8月	橋爪宏，橋本登
TM-113 リフトジェットエンジン試験設備（II）	1967年9月	森田光男，岩部柱相 関根静雄，武田克巳
TM-114 五段遷音速軸流圧縮機の空力設計	1967年9月	藤井昭一，松木正勝 五味光男
TM-115 燃料蒸発管に関する研究（I）	1967年9月	大塚貞吉，鈴木邦男 田丸卓，乙幡安雄
TM-116 高負荷燃焼器の空気孔からの流れについて（I）	1967年9月	鈴木邦男，相波哲朗
TM-117 ロケット用テレメータ機上装置の集積回路化	1967年9月	新田慶治，松崎良継
TM-118 操縦桿レート信号によるアイアンバード制御の安定効果とパイロットのモデルについて	1967年9月	村上力，真柳光美
TM-119 ベーン型気流方向検出器の特性	1967年11月	田畑浄治，松島弘一 成田健一，塚本憲男
TM-120 円錐管レンズの設計とその応用	1967年11月	山中竜夫，奥岨澄男

注：欠番は配布先を限定したもの

航空宇宙技術研究所資料121号

昭和42年11月発行

発行所　航空宇宙技術研究所
東京都調布市深大寺町1880
電話武蔵野三鷹(0422)44-9171（代表）

印刷所　一誠社綜合印刷株式会社
東京都武蔵野市御殿山1-6-10

Printed in Japan