講　座

スパース行列処理技法（3）*

大附辰夫** 川北建次**

7. コンピュータ・アーキテクチャとプログラミング技法

行列処理を主体とする多くの応用プログラムでは，スパース行列処理がその性能を決定する大きな要素となっている．したがって，スパース行列処理プログラムの設計と作成にあたっては，性能を決める種々の要因とそれらの関係に十分注意する必要がある．性能を決定する最も基本的な要素は，アルゴリズムとデータ構造であるが，実際にスパース行列処理プログラムを作成する場合は，計算機のアーキテクチャ，言語，コンパイラー，OS などの機能と性能を考慮し，問題の性格に適した設計を行わなければならない．

本章では，スパース行列処理の性能とそれに関連する計算機のアーキテクチャ，OS，プログラミング技法について述べ，最後に，スパース行列処理をより効率的に行うための新しいアーキテクチャの問題について説明する．

7.1 性能評価

最も基本となる評価パラメータは演算量とデータ量である．演算量はふつう単位演算を掛算として問題の大きさnの多項式として表わされる．データ量は1つのデータ（ワード）を単位として同様の多項式で表わされる．これらの評価を基本として，実際に性能をあらわすパラメータとしてはふつう CPU 時間，メモリー量，I/O 回数などが使われている．

主記憶だけで処理を行う場合は，CPU 時間，メモリー量は問題の大きさの関数（多項式）となるが，I/O を行う場合は，CPU 時間，I/O 回数は問題の大きさとメモリー量の関数として表わされる．仮想記憶機能をふくめて一般に I/O を行う場合は，CPU の演算能力だけでなく OS のメモリー管理機能，データ管理機能，チャネル，外部記憶装置などが複雑に関連しあうため，それらの要因を明確に評価することは容易ではない．

7.2 プログラミング技法

プログラミングは，使用する計算機のアーキテクチャ，使用言語，コンパイラー，OS と密接な関連をもって行われるので，計算機システムごとにそれぞれ特異性をもつことは避けられないが，ここでは，最近の計算機システムがもつ代表的な機能を仮定して話を進めたい．

設計上まず問題となるのは CPU 時間とメモリー量のバランスである．スパース行列処理プログラムに課せられた目標に従ってバランスが決定され，必要ならば I/O を行うことになるが，一般に I/O を行うと効率は大幅に低下するのが普通である．したがって，できるだけ主記憶内で処理することをまず考えるべきであろう．最近は大容量の主記憶を備えた計算機が多くなっており，メモリーの LSI 化にともなって増々大容量化する傾向にあるので，主記憶内で処理できる問題の規模は増大していくと考えられる．

プログラミングでまず問題となるのはデータへのアクセスである．スパース構造をあらわすデータ構造は，いくつかの配列または構造体配列によって構成されている．配列要素のアクセスは通常，配列の添字を与えられてそれを絶対アドレスまたは相対アドレスに変換してアドレッシングが行われている．したがって，アドレス計算の回数をできるだけ少なくするような処理手順とデータ構造の構造化（まとめること）を行うことが望ましい．また，メモリーのアクセスを効率化するためには，できるだけ連続した場所を読み出すようにすべきである．データ構造の構造化はメモリーアクセスの効率化の面でも有効である．メモリーのアクセスはきまったワード数（ワード幅）を単位として行われるので，ワード幅よりも短いデータが2つのワードにまたがらないようにデータのマッピングにも注意

* Sparse Matrix Technology by Tatsuo OHTSUKI and Kenji KAWAKITA (Central Research Laboratories, Nippon Electric Co., Ltd.)

** 日本電気(株)中央研究所

する必要がある．次に注意すべきことは，プログラムとデータの分離である．作成上，プログラムとデータを明確に分離することによって，より効率的な開発ができるだけでなく，実行能率を高める点でも効果があるからである．通常，主記憶装置は独立にアクセスできるいくつかのメモリーユニットで構成されている．CPU からは主記憶内にあるデータと命令の読み出しが行われるが，とり出すべき命令とデータが同じメモリーユニットにある時は，命令のとり出しとデータのとり出しで競合が起こり，実行能率が低下することになる．したがって，プログラムとデータを分離してメモリー上におくことによって，メモリーアクセスの競合を少なくすることができる．

最近の計算機は何重ものメモリー階層(例えば，バッファメモリー，メインメモリー，大容量ディスク)をもち，仮想記憶機能を提供しているものが多い．ユーザは実メモリーの大きさやメモリー階層を意識することなく，大きいアドレス空間で処理を行うことができる利点がある反面，メモリー階層間のデータの転送を制御できないために，プログラムの設計のしかたと使用法によって性能に大きい差が出ることがある[131)~134)]．メモリー階層間のデータは，ページあるいはセグメントと呼ばれる大きさを単位とし，高い階層に必要とするデータがない時は，下の階層から転送が行われる．したがって，ひんぱんに転送が行われると実行能率は大幅に低下することになる．したがって仮想記憶機能を使う場合は，アルゴリズムとデータ構造の設計でメモリー階層の管理のしかたに十分注意する必要がある．一般的な指針としては，前述したようなデータ構造の構造化は有効であろう．**図-19**(a)は構造化をしない場合，図-19(b)は構造化した場合の例であるが，配列の i 番要素をとり出すのに前者では3回のアドレス計算が必要である上，各配列 A, B, C は異なるページにある確率が高く，メモリー階層間の転送量は多くなる．一方，後者では1回のアドレス計算で良いだけでなく，各データが連続しているので同一ページにあると考えられ，前者に比べて能率はずっと良い．

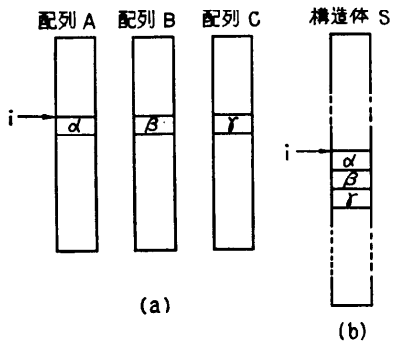

図-19 データの構造化

最近の大型高性能計算機では，パイプライン制御により命令の並行処理を行うことによって実行能率を上げているものがある[135),136),144)]．パイプライン制御の計算機は独立に実行できるいくつかの演算プロセッサ(例えば，固定小数点演算，浮動小数点演算，文字処理の各プロセッサ)をもち，命令列を独立に実行できる命令に分解して並行処理を行うものである．スパース行列処理の場合は，アドレス計算のための固定小数点演算と浮動小数点演算が主たる演算である．この場合のプログラミングにおける一般的な指針は，レジスタをできるだけ有効に使うこと，同じタイプの命令を隣りあわないようにすることである．例えば，消去法における代表的な演算である $A(I)=A(J)-A(K)/A(L)$ の場合は，浮動小数点演算の間にアドレス計算のための固定小数点演算を挿入することと，隣り合う命令で同じレジスタをできるだけ使わないようにすることがポイントとなる．パイプライン制御の計算機は，通常，数十個の命令が入る命令バッファを持っており，命令バッファ中に入るような演算ループの場合は命令バッファ中でループの演算が行われ（ループモードという.)，命令列の読み出し，解読が不要となるので，非常に高速の演算ができる．したがって，最も深い演算ループがループモードで実行できるように設計できれば，演算ループを展開してループのネストを浅くした場合以上の効率が期待できる．

次に，I/O を行う場合の問題について述べる．I/O を行う場合，I/O の回数をできるだけ少なくする事が設計上のポイントである．一般に，READ あるいは WRITE マクロが出されると，OS のデータ管理機能によりチャネルプログラムの設定と実行のための準備が行われ，その後チャネルに制御がわたされてデータの転送が行われる．マクロを実行するためには，CPU による処理がかなり必要であり，オーバヘッドとなる．I/O 回数を少なくするためには処理順序とデータ構造，レコードの設計が問題となるが，一般的な指針としては，1つのレコードを参照したときに行われる演算の量をできるだけ多くすることである．この他に，I/O に関連した問題としては，バッファの割り付け位置，バッファの数，チャネルプログラムの設計などがある．一般に，CPU によるメモリーアクセスと

チャネルによるメモリーアクセスが同一メモリーユニットで行われると競合が起こり，CPU の実行能率を低下させる．したがって，バッファは，CPU の処理と競合が起こりにくい位置に割り付ける必要がある．多重バッファを使って，先の計算で必要なデータを先読みすることは，総処理時間を短くする点で効果がある．I/O の回数を実効的に減少させる方法として，チャネルプログラムのコマンドチェイニングを利用する方法がある．これは，1回のI/O要求で複数個のレコードを処理するものである．つまり，1度に処理されるレコードごとに，バッファとチャネルプログラムを用意し，1つのチャネルプログラムの実行が終了した時点で，CPU に制御をもどさないで次のチャネルプログラムを実行させる方法である．さらに高度の技法としては，チャネルのプログラム制御割り込みによる方法がある．これは，チャネル制御語がチャネルに受入れられた時点でチャネルから CPU に一種の割り込みをかけ，CPU 側からチャネルプログラムの変更を動的に行う方法であり，より高度の処理が可能となる．

7.3 スパース行列処理指向のアーキテクチャ

現在，実行能率を向上させるためのアーキテクチャとして，パイプライン[135),136)]，マルチプロセス，パラレルプロセス[136)-143)]，アレイプロセッサ[144),145)]などがあるが，スパース行列処理を特に意識したアーキテクチャは非常に少ない．実在するプロセッサに組み込まれているスパース行列処理向きの命令は，**図-2C** のように2つのインデクス配列でさされるデータの演算である（例えば内積演算）[144)]．勿論これだけでは不十分であり，将来の発展が期待されるわけであるが，LSIやマイクロプログラム技術の進歩により，実現性はあると考えられる．スパース行列処理向きのアーキテクチャを設計する上でのポイントは，従来の命令列を1つにまとめた複合命令を設計し，並行処理を組み合わせることであろう．そのためには，行列処理における基本的な演算を抽出する必要がある．消去法あるいは分解法における主な演算は，次の3種類に分類できる．

$$\begin{cases} a=a-c*d \\ a=r*(a-\alpha^T\beta) \\ \alpha=\alpha-a\beta \end{cases}$$

スパース行列を表現するいくつかの基本的なデータ構造に対し，上述の命令を定義し，各命令の実行に並行処理を入れることにより，非常に高速の演算が可能となる．したがって，大規模な問題を処理するために将来非常に有力な方法となるであろう．

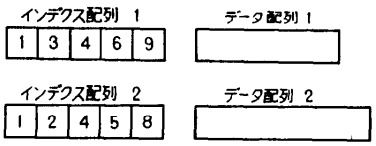

図-20 インデクス配列で与えられるデータ

* 既約接続行列 $D=\{d_{ij}\}$ は，ある基準点を定め，それ以外の節点 $i=1,2,\cdots,n$ について，
節点 i が枝 j の始点のとき $d_{ij}=1$,
節点 i が枝 j の終点のとき $d_{ij}=-1$,
その他の場合 $d_{ij}=0$
となるように定義された行列である．

8. 応　用

スパース行列処理技法は幅広い応用分野を持つが，本章では代表的なものをいくつか取り上げて，その中でのスパース構造を持つ方程式の役割について論じてみたい．前章までに，スパース行列処理のための標準的なアルゴリズムやプログラミング手法については紹介したので，本章では応用分野ごとにその中に介入するスパース行列の特徴や，それに応じて要求される個有の処理技法などについて概略を解説する．本章では，電気回路解析，構造解析，線形計画法，偏微分方程式の数値解法を特にとりあげて説明するが，流体，電力の問題などについては文献 170)～174) を参照されたい．

8.1 電気回路解析

電気回路網を構成する素子を枝に，等電位点を節点に対応させれば，その接続関係は有向グラフによって表現できる．ここで枝の方向は「電流は指定された方向に流れるとき正の値をとり，逆方向に流れるとき負の値をとる」という意味をもつもので，枝の方向の与え方は任意である．電気回路網の基本方程式は，**キルヒホフの電流則**と**電圧則**および**素子特性**（線形の場合は**オームの法則**に相当する．）とから成る．この内，キルヒホフの両法則は，電気回路網を抽象化したグラフの接続関係だけから定まる．接続関係は通常，**既約接続行列***と呼ばれる $n\times m$ の行列 D で表現される．ここで，$(n+1)$ は節点の数，m は枝の数である．キルヒホフの電流則は，「各々の節点に流入する電流の総和は0である(**図-21**(a)次頁参照)」という関係で，形式的に

$$Di=j \qquad (8.1)$$

と記述できる．ここで i は枝電流を表わす m 次の列ベクトル，j は外部から与えられた電流源をあらわす n 次の列ベクトルである．キルヒホフの電圧則は，「各々の閉路に含まれる枝に沿った電圧の総和は0である

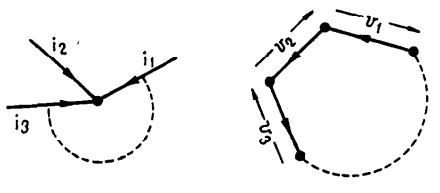

図-21　キルヒホフの電流則(a)と電圧則(b)

(図-21(b)参照)」という関係で,

$$D^T\boldsymbol{u}-\boldsymbol{v}=\boldsymbol{e} \qquad (8.2)$$

と記述できる*. ここで $\boldsymbol{u}$ は基準点から見た節点電位を表わす n 次の列ベクトル, $\boldsymbol{v}$ は枝電圧を表わす m 次の列ベクトル, $\boldsymbol{e}$ は外部から与えられた電圧源を表わす m 次の列ベクトルである.

素子特性はグラフの構造とは無関係なもので, 各々の枝（素子）の物理的な特性によって定まる. 素子には**図-22** に示すように基本的に3種類あり, それぞれの特性――まず線形と仮定する――は次のようになる

- 枝 k の電流, 電圧の関係が

$$i_k=G_kv_k \qquad (8.3)$$

という形式で与えられるとき, 枝 k は**抵抗**と呼ばれる.

また, 電流, 電圧の関係が

$$\begin{cases} v_k=d\varphi_k/dt & (8.4) \\ \varphi_k=L_ki_k & (8.5) \end{cases}$$

という形式で与えられる枝 k を**インダクタ**,

$$\begin{cases} i_k=dq_k/dt & (8.6) \\ q_k=c_kv_k & (8.7) \end{cases}$$

という形式で与えられる枝 k を**キャパシタ**と呼ぶ.

以上の方程式を連立させれば, 形式的に,

$$A\boldsymbol{x}=\boldsymbol{b} \qquad (8.8)$$

と記述でき, $\boldsymbol{b}$ を与えて式(8.8)を満たす $\boldsymbol{x}$ を求める問題が回路網解析である. 係数行列 A のスパース構造を図示すると**図-23** のようになる. ここで D は非零要素として“1”または“−1”だけを含むスパース行列, G,L,C は正値対角行列, $(d/dt)I$ は微分オペレータから成る対角行列であり, 電気回路解析の能率は, このようなスパース構造を如何に有効に利用するかにかかっている.

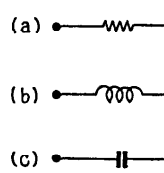

図-22　電気回路網を構成する素子: 抵抗(a), インダクタ(b)およびキャパシタ(c)

以上の議論で全ての素子が線形と仮定したが, 非線形素子が含まれる場合は, 対角行列 G,L,C の要素のいくつかが定数ではなく非線形オペレータとなる. 実際の数値解析では, 非線形オペレータの位置に, ある動作点における偏微係数の値が代入されたスパース行列を取り扱うことになる. ここで非線形素子が含まれる場合でも, 一部の非零要素の値は変化するが, そのスパース構造は変化しないことが重要である. また, トランジスタのような半導体デバイスを含む場合, 一つの素子が式(8.3)～(8.7)のように一つの枝について電流, 電圧の関係式で与えられるとは限らず, 一般に二つ以上の枝が一つの素子を構成することになる. この場合, G,L,C などは対角行列になるとは限らず, また, 対称性もくずれるが, 対角行列に近い極めてスパースな行列であることには変りはない.

電気回路解析の中で, 基本的なものは**直流解析**, **過渡解析**, **交流解析**の三つがある. 直流解析は静止状態を論ずるもので, $d/dt=0$ すなわち, 全てのインダクタの電圧および全てのキャパシタの電流を0とおいた時の抵抗だけから成る回路を解く問題となる. 数学的にいえば, 直流解析は, 全ての抵抗が線形であるか否かによって, 線形または非線形の連立方程式を解く問題である. 過渡解析は, 各枝の電流, 電圧の値の時間的変化を対象としたもので, 式(8.8)全体で記述され

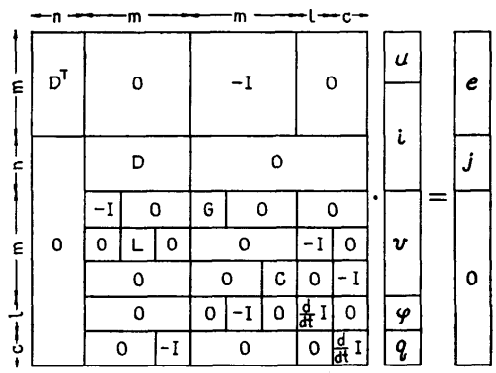

図-23　電気回路網の基本方程式のスパース構造
m: 枝の数, $n+1$: 節点の数, e: インダクタの数, c: キャパシタの数

* ある枝 j の電圧を v_j, 始点, 終点の電位をそれぞれ $u_j^{(+)}, u_j^{(-)}$ とすれば, $v_j=u_j^{(+)}-u_j^{(-)}$ であり, この関係を全ての枝 $j=1,2,\cdots, m$ について記述した式(8.2)はキルヒホフの電圧側と等価である.

る線形または非線形の常微分方程式を解く問題である．微分オペレータ d/dt の扱いは，数値積分法として何を採用するかに依存する．例えば後退オイラー法[115),145)]を適用すれば，式(8.4)，(8.5)のインダクタの特性は

$$v_k=(L_k/\Delta t)\cdot i_k-\varphi_k^{(0)}/\Delta t \qquad (8.9)$$

と近似される．ここで $\varphi_k^{(0)}$ はある時刻 t における φ_k の値で，すでに計算が終了しているものと仮定する．また，v_k, i_k は時間きざみ Δt 後の v, i の値で，これから求めようとする変数の値である．同様にして，式(8.6)，(8.7)のキャパシタの特性は，

$$i_k=(C_k/\Delta t)\cdot v_k-q_k^{(0)}/\Delta t \qquad (8.10)$$

と近似される．式(8.9)，(8.10)を図示すれば，それぞれ**図-24**(a)，(b)のようになり，いずれも抵抗と電流の組み合わせによって表現できる．従って，過渡解析も線形または非線形の連立方程式を時間きざみごとに繰り返して解く問題となる．交流解析は線形回路に対してだけ意味をもつもので，これを正弦波電源で駆動した時の周波数特性を求める問題である．これに対しては，交流理論における演算子法[147)]が適用でき，微分オペレータを

$$d/dt=j\omega;\ j=\sqrt{-1} \qquad (8.11)$$

とおくことによって，複素数空間における線形方程式をいくつかの角周波数 ω について繰り返し解く問題と考えられる．以上の基本的な三種類の回路解析の各々について，いくつかの素子の値が変化した時に各部の変数がどのように変化するかという問題を論ずる**統計解析**がある．この場合は，G, C, L などの要素をパラメータとみなして，上記の基本的な解析を更に繰り返すことになる．以上説明したように，回路解析とは，スパース構造が一定な線形連立方程式を多数回繰り返して解くことである．

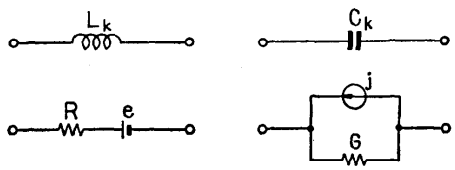

図-24　インダクタ特性(a)およびキャパシタ特性(b)の後退オイラー法による近似

さて，図-23 の構造を持つ方程式(8.8)をながめると，数値解析を行わなくても，かなりの部分を消去して方程式の元数が減らせることがわかる．回路のグラフ構造だけに着目して処理し得る部分を予め消去してしまって連立方程式の元数を減らして数値計算を行うためのいくつかの典型的な定式化手法が確立されている．代表的なものとして，**節点解析，カットセット解析，閉路解析**などがあるが，これらの内容については，例えば文献 148)，149) を参照されたい．一方，最近は図-23 の構造，あるいは極く一部を変形しただけの大きな行列——スパースタブローという——をそのまま扱うことによって問題に応じて最も能率の良い計算順序を導くことを狙った解析法が用いられるようになって来た．この方法は **6.** で述べた**コード発生方式**を利用するのに適しており，新しい汎用回路解析プログラムのいくつかに組み込まれている[114)~116)]．

電気回路解析は，大規模スパース行列を扱うという意味では他の応用分野と変わらないが，

ⅰ) 半導体素子の出現により，正値対称行列だけを対象とした解析法が役に立たなくなった．

ⅱ) スパース構造が不規則である．

ⅲ) 同じスパース構造を持つ行列に対して，極めて多数回の解析が要求される．

というのが，構造物，電力系統，流体輸送網などの分野とは異なった特徴であろう．電気回路解析の分野の中で考案されたコード発生方式は，上記の特徴を背景としていると考えられる．しかし，原理的に他の分野に応用できる要素も多いので，今後多くの分野に浸透していくことが予想される．

8.2 構造解析

構造解析は，建築，土木，機械などの構造物の力学的特性を解析する分野である．構造物の力学的特性をあらわす基本的な物理量は，構造内の応力，変位であり，構造物を小さい要素 (finite element) に分割し，要素間の物理量の関係式を作ることにより解析が行われる．基本的な定式化手法として，応力を独立変数とする**応力法**と，変位を独立変数とする**変位法**がある*．普通，応力法で作られる連立一次方程式の次数は，変位法によるものよりも小さいので，変位法よりも優れている点はあるが，構造解析では大規模な構造物や複雑な形状の構造物を対象とすることが多く，簡単な処理で方程式を作ることが必要である．上述の点で変位法は応力法に比べてはるかに簡単に方程式を作ること

* 応力と変位の両方を独立変数とする混合法もあるがほとんど使われていない．

ができ，かつ計算機による処理も容易である．最近の構造解析プログラムでは，連続体を分割して解析する統一的な手法である**有限要素法** (finite element method[150]) と，変位法を組み合わせた処理を行うものが多い[124),125)]．以下に変位法と有限要素法にもとづいた構造解析の手法と行列処理の問題について述べる．

まず，構造体を比較的単純な形状の要素（例えば，2次元の場合は三角形）で分割する．各要素には節点が対応し，各節点ごとに変位を独立変数とする方程式を作ることができ，結局

$$K\boldsymbol{x}=\boldsymbol{p}$$

の形の連立方程式が得られる．$\boldsymbol{x}$ は各節点の変位から成るベクトル，$\boldsymbol{p}$ は外部から与えられた力のベクトルである．係数行列 K は，**スティフネス行列** (stiffness matrix) と呼ばれる．行列 K は対称かつ正定値であり，実際に解析される問題ではスパースとなっている．K は普通帯行列となることが多いが，帯の幅は構造物の形状，節点の番号付けの方法により異なる．変位法の場合，節点の順序付けは，構造を与えるグラフにおける節点消去法の問題と同じであることに注意されたい．

構造解析では，限られた範囲の問題に対し，SOR法などの逐次解法が有効に使われたことはあるが[151)]，現在ではほとんど直接解法が使われている．逐次解法の収束性は，構造物の形状，境界条件，材質の構成などに強く依存しており，汎用構造解析プログラムには適さないことだけでなく，いろいろな荷重に対する解を求めることが多いからである（直接解法では，異なる荷重に対する解は若干の手間の増加で計算ができる）．

大規模な構造物を解析する場合，構造物をいくつかの小さい構造に分割して解く方法は非常に有効である．一般的な分割解法としては，Kron の方法[152)]が有名であるが，処理が複雑なためあまり使われていない．Iron による分割解法[124),153)]は，分割された構造体ごとにスティフネス行列を求め，それをもとにして全体の構造の解析を行う方法で，少ないメモリーで効率の良い分割解法が可能である．

構造解析における連立一次方程式の解法では，帯行列処理にもとづいたスパース行列処理が行われることが多い．また，構造物の形状が複雑になると帯の内側に不規則性があらわれるため，ふつう帯の内側に対してもスパース行列処理を行う[110),154)]．

8.3 線形計画法

線形計画法は最も基本的で，かつ広い応用分野を持つ数理計画法であり，過去数十年の間に，手法の改良の面においても，応用分野の拡大の面においても，めざましい発展をとげて来た．最近では大型計算機を用いれば，数万の式から成る問題も解けるといわれている．この発展の要因の半分は計算機ハードウェアの進歩によっているが，残りの半分はアルゴリズム及びプログラミング技法——特にスパース行列処理技法——の進歩によっている．

線形計画の問題は，一般性を失うことなく*

$$\begin{cases} x_0-\sum_{j=1}^{m} c_j x_j=0 \\ \sum_{j=1}^{m} a_{ij}x_j=d_i;\ i=1,2,\cdots,n \\ x_j\geqq 0;\ j=1,2,\cdots,m \\ x_0\rightarrow\min \end{cases} \quad (1)$$

という標準形で記述することができる．ここで，$m>n$，かつ冗長な式は含まれていない，すなわち係数行列 $A=\{a_{ij}\}$ の階数は n と仮定する．A の m 個の列の中から適当に n 個の線形独立な列を取り出して得られた非特異行列を B，残りの $(m-n)$ 個の列から成る部分行列を R で表わせば，式(1)の上の二つの式は，

$$\begin{cases} x_6-\boldsymbol{b}^T\boldsymbol{u}-\boldsymbol{r}^T\boldsymbol{v}=0 & (2) \\ B\boldsymbol{u}+R\boldsymbol{v}=\boldsymbol{d} & (3) \end{cases}$$

という形式で記述できる．ここで $A=[B\,|\,R]$ という分割に対応して $\boldsymbol{c}=\{c_1,c_2,\cdots,c_m\}^T$ および $\boldsymbol{x}=\{x_1,x_2,\cdots,x_m\}^T$ も

$$\boldsymbol{c}=\begin{bmatrix}\boldsymbol{b}\\ \cdots\\ \boldsymbol{r}\end{bmatrix},\quad \boldsymbol{x}=\begin{bmatrix}\boldsymbol{u}\\ \cdots\\ \boldsymbol{v}\end{bmatrix}$$

と分割表示されている．この分割において，$\boldsymbol{u},\boldsymbol{v}$ はそれぞれ B に対応した**基底変数** (basic variable)，**非基底変数**と呼ばれる．式(3)の基底変数 $\boldsymbol{u}$ を非基底変数 $\boldsymbol{v}$ で表わし，更にこの結果を式(2)に代入すれば

$$\begin{cases} x_0-\boldsymbol{\gamma}^T\boldsymbol{v}=\alpha & (2)' \\ \boldsymbol{u}+S\boldsymbol{v}=\boldsymbol{\beta} & (3)' \end{cases}$$

という正規化表現が得られる．但し，

$$\begin{cases} \boldsymbol{\gamma}^T=\boldsymbol{b}^TB^{-1}R-\boldsymbol{r}^T,\ \alpha=\boldsymbol{b}^TB^{-1}\boldsymbol{d} \\ S=B^{-1}R,\ \boldsymbol{\beta}=B^{-1}\boldsymbol{d} \end{cases} \quad (4)$$

である．式(2)′,(3)′の正規化表現は，$\boldsymbol{v}=\boldsymbol{0},\boldsymbol{u}=\boldsymbol{\beta}$ とおいた時に目的関数 x_0 は値 α を取る，ということを表わしている．一般に A の n 次の非特異行列 B に対

* 一般には，非負条件以外にも不等式がある場合，一部の変数に非負条件がない場合などいろいろあるが，いずれも式(1)の形式に帰着できることはよく知られている．

して非基底変数を0と置いたときのxを**基底解** (basic solution) と呼ぶ. Dantzing によって提案された**シンプレクス法**[155] (simplex method) は，現在線形計画問題の解法の基本となっているが，これは「少なくとも一つの基底解は最適解である」という事実に基づいている.

シンプレクス法の実行はxの係数および右辺を行列形式で表示したもの——**シンプレクス・タブロー** (simplex tableau) と呼ばれる——基底の変換に対応して次々に変更していく操作を基本としている. 式(2),(3)に対応するタブロー $\hat{A}$ は**図-25**(a)の形で，式(2)′,(3)′に対応するタブロー T は同図(b)の形で表わされる. ここで

$$\hat{B}=\left[\begin{array}{c|c}1 & -\boldsymbol{b}^T \\ \hline \boldsymbol{0} & B\end{array}\right] \qquad (5)$$

とおけば，その逆行列は

$$\hat{B}^{-1}=\left[\begin{array}{c|c}1 & \boldsymbol{b}^T B^{-1} \\ \hline \boldsymbol{0} & B^{-1}\end{array}\right] \qquad (6)$$

となり，更に

$$T=\hat{B}^{-1}\hat{A} \qquad (7)$$

という関係が成立することに注意されたい. 簡単のため，全ての $i=1,2,\cdots,n$ に対して $\beta_i \geqq 0$ となるような基底 B が与えられている*と仮定する. この時，$\boldsymbol{v}=0$ とおいた $\boldsymbol{x}$ は全ての制約条件を満たすことから，**主実行可能** (primal feasible) である，といわれる. また，全ての $j=1,2,\cdots,m-n$ に対して $\gamma_j \geqq 0$ となれば，その基底は**双対実行可能** (dual feasible) である，といわれる.「主実行可能かつ双対実行可能な基底解は最適解である」という事実はよく知られており，シンプレクス法のフェースⅡ (Phase Ⅱ) とは，主実行可能であるが，双対実行可能とは限らない基底解から出発して，主実行可能という性質を保ちながら目的関係 $x_0=\alpha$ を小さくするような基底変換を繰り返し，最終的に双対実行可能な解，すなわち最適解を求める手順である.

さて，シンプレクス法の一回の基底変換の操作の骨組を記述すると以下のようになる**.

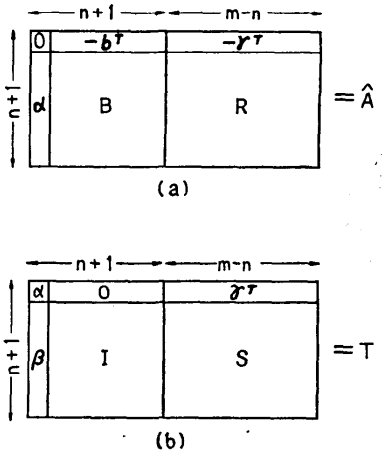

図-25 シンプレクス・タブローの原形(a)と基底変換後の構造(b)

①：新しく基底に操入すべき A の列 $\boldsymbol{a}^{(k)}$ を

$$\gamma_k=-\max_{j;\gamma_j<0}|\gamma_j| \qquad (8)$$

によって決定する***.

②：$\boldsymbol{a}^{(k)}$ に対応した S の列 $\boldsymbol{A}^{(k)}$ を取り出す.

③：基底から除くべき A の列 $\boldsymbol{a}^{(l)}$ を

$$\frac{\beta_l}{s_{lk}}=\min_{i;s_{ik}>0}\left[\frac{\beta_i}{s_{ik}}\right] \qquad (9)$$

によって決定する****.

④：新しい基底に基づいてタブローを変換する，すなわち，新しい α,β,γ および S を求める.

シンプレクス法が最初に考案された時は，上記のステップ④において，実際に s_{lk} に関する枢軸変換を行うことによって新しい基底に対応するタブローを陽に求めるという方法が採用されていた. ところが，大規模な問題においては，基底変換を行うたびにタブローの中に新しい非零要素がどんどん発生し，演算回数の面でも，メモリー容量の面でも望ましくないことが認識されるようになった. そこで，基底に対応したタブローを陽に求めずに，$\hat{B}^{-1}$ だけを計算しなおし，実際に必要であるところの T の一行目，一列目および k 列目だけを $\hat{B}^{-1}$ と原形のタブロー $\hat{A}$ から計算するという方法——**改訂シンプレクス法** (revised simplex method) という——が考案された. また，B^{-1} も陽に求めると全ての要素が非零になってしまうので積型逆行列 (**2.4** 参照) によって陰的な表現をする方法が多く用いられるようになった[23)~25)]. 更に最近では，積型逆行列よりもスパース性を保存できる消去形逆行列 (**2.1** 参照) が用いられるようになって来た[22)].

---

*一般には，このような基底を求めるために予備的な線形計画問題——フェース I (Phase I) の操作と呼ばれる——を解かねばならない.

**このアルゴリズムの正当性を実証するための理論，およびアルゴリズムを遂行する上で介入する退化 (degeneracy)，巡回 (cycling)，丸め誤差などの問題については省略する. これらについては適当な参考書類 (文献23),157),158) など) を参照されたい.

***全ての j に対して $\gamma_j \geqq 0$ ならば $x_0=\alpha$ は最小値である.

****全ての i に対して $A_{ik} \leqq 0$ ならば，有限の最小値は存在しないことを示す. また，$\beta_l=0$ ならば退化が起こっていることに相当する.

$\hat{B}$ の逆行列の陰的表現が与えられた場合に，ステップ②および④を実行する手順は次のようになる．

②：S の k 列目

$$s^{(k)}=B^{-1}a^{(k)} \tag{10}$$

は B^{-1} の陰的表現の因数となる各行列と列ベクトルとの積演算のくり返し——**前進変換** (forward transformation) という——によって得られる．

④-i)：T の k 列目を $(\gamma_k, s_k, \cdots, s_{lk}, \cdots, s_{mk})^T$，$\hat{B}$ の列 $\hat{a}_l$ を新しい列 $\hat{a}_k$ で置き代えた基底行列を $\hat{B}'$ とすれば，新しい基底逆行列は

$$(\hat{B}')^{-1}=J_l\hat{B}^{-1} \tag{11}$$

によって与えられる．ここで J_l はその l 列目が $(p_0, p_1, \cdots, p_m)$ となっている他は単位行列と同じ構造（すなわち図-2(b)の構造）を持った行列である．但し，

$$\begin{cases} p_l=1/s_{lk} \\ p_0=-\gamma_k\cdot p_l \\ p_i=-s_{ik}\cdot p_l\ ;\ i\neq l \end{cases} \tag{12}$$

である．

④-ii)*：T の1列目 $\hat{\beta}$ は $\hat{A}$ の1列目 $\hat{d}$ を用いて

$$\hat{\beta}=\hat{B}^{-1}\hat{d} \tag{13}$$

と表現でき，②と同様の前進変換によって α および β が求められる．

④-iii)：T の1行目の非基底部分 γ^T は長さ $(n+1)$ の行ベクトル $e^{(1)}=(1, 0, 0, \cdots, 0)$ と $\hat{A}$ の非基底部分 $\hat{R}$ を用いて

$$\gamma^T=e^{(1)}\hat{B}^{-1}\hat{A} \tag{14}$$

と表わされる．列ベクトルと行列の積演算の繰り返し——**後退変換** (backward transformation) という——によってその値が求められる．

シンプレクス法における初期段階の基底逆行列が消去型逆行列による表現で与えられているとして，上記のアルゴリズムを p 回反復した時の基底逆行列は，

$$\hat{B}^{-1}=J_pJ_{p-1}\cdots J_1\ (U_2\cdots U_{n+1}, L_{n+1}, \cdots L_1) \tag{15}$$

という形式で与えられていることになる．現在使われている殆んどの汎用の線形計画パッケージでは式(15)の表現を用いているが，$J_1, \cdots, J_p$ の中に発生する非零要素が多すぎるのが欠点である．最近では，**2.5** の式 (2.34) の表現や，これを更に改良した方法[26)~31)]なども利用されるようになってきた．

上記の行列のスパース性を利用した手法によって，より大きな問題が主記憶だけを用いて解けるようになったことはいうまでもないが，更に大きな問題に対して主記憶と二次記憶とのデータの交換を能率良く行わせる上でも，スパース行列手法は重要な役割を演じている．上記のアルゴリズムでは，行列の要素は常に列方向に参照されるので，例えば非零要素およびそのインデクスを列ごとに行番号順に取り出し，これを一次元配列の形式に記憶するというデータ構造（**5.** の行／列インデクス方式参照）が適している．

シンプレクス法の処理時間（CPU 時間+I/O 時間）と主記憶の大きさとの関係は，定性的には**図-26** のような傾向をもつので，大規模な線形計画問題を解くには，ハードウェアの選択においても，プログラミング技法の設計においても，いかに I/O 時間を少なくするかが重要な問題となる．

8.4 偏微分方程式の数値解法

偏微分方程式は，拡散，波動，熱伝導，原子炉，大気汚染など多くの科学，工学の分野にあらわれ，数多くの解析が行われている．基本的には，偏微分方程式に差分公式**を適用して得られる差分方程式を解く問題となる．差分方程式の組をまとめると，結局，連立一次方程式となるが，一般に偏微分方程式の差分化によって得られる連立一次方程式は大型かつ規則的なスパース性のある係数行列をもつ．また，対称かつ対角優勢など数値的に良い性質をもっていることが多い．数値的に望ましい性質をもつことと，スパース行列処理の容易さのため，多くの問題で逐次解法が使われ，さまざまな逐次解法が開発されている***．偏微分方程式の分野では，現在も逐次解法で解かれる問題は非常に多いが，逐次解法は領域の形状や境界条件が複雑になると収束しにくい欠点があり，信頼性に欠ける．このため，最近，直接解法を使って解く方法が開発されてきている[162)~166)]．また，逐次解法と直接解法を組み合わせた方法もある[167)~168)]．

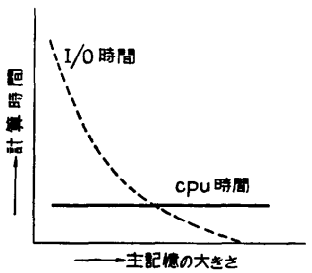

図-26 主記憶の大きさとシンプレクス法の計算時間の関係

* 以下新しい基底行列 $\hat{B}'$ をあらためて $\hat{B}$ とおく．

** 偏微分方程式のタイプにより適用する差分公式が異なる．差分公式については文献 169) などを参照されたい．

*** 逐次解法については文献 159)~161) を参照されたい．

9. おわりに

スパース行列処理技法の基礎理論，アルゴリズム，プログラミング技法，応用などに関する最近の技術の動向を3回にわたる連載により解説してきた．基礎的な事項だけでなく，新しい研究成果もなるべく盛り込むように努力したが，筆者らの力不足のために，不適当な判断や重要な事項の見落としなどもあるのではないかと思われるが御容赦いただきたい．また，スパース行列処理技法は現在幅広く研究されており，我々をとりまくシステムの大規模化に伴い，今後もますます重要となっていく課題であるので，新しい研究成果によって本講座の内容が近い将来陳腐なものとなってしまうということも十分考えられる．その時は，適当な専門家の方に再びこの課題を解説して戴くのも一案であろう．

参考文献

(既出分への追加)

131) J.E. Morrison: User Program Performance in Virtual Storage Systems, IBM Syst. J., Vol. 12, pp. 216～237 (1973).

132) A.C. Mckellar and E.G. Coffman: Organizing Matrices and Matrix Operations for a Paged Memory Systems, Comm. ACM, Vol. 12, pp. 153～165 (1969).

133) P.J. Denning: Virtual Memory, Comput. Surveys, Vol. 2, pp. 155～189 (1970).

134) R.L. Matton ***et. al.***: Evaluation Techniques for Storage Hierarchies, IBM Syst. J., Vol. 9, pp. 78～117 (1970).

135) T.C. Chen: Parallelism, Pipelining and Computer Efficiency, Comput. Design, pp. 69～74 (1971).

136) IBM System/360 and System/370 Model 195 Functional Characteristics, GA 22-6943.

137) M.C. Pease: Matrix Inversion using Parallel Processing, J. Assoc. Comput. Mach., Vol. 14, pp. 757～764 (1967).

138) G.C. Sedler: Parallel Numerical Methods for Solution of Equations, Comm. ACM, Vol. 10, pp. 286～290 (1967).

139) G.H. Barnes ***et. al.***: The ILLIAC IV Computer, IEEE Trans. C-17, pp. 746～757 (1968).

140) D.J. Kuch: ILLIAC IV Software and Application Programming, IEEE Trans. C-17, pp. 758～770 (1968).

151) W.L. Miranker: A Survey of Parallelism in Numerical Analysis, SIAM Rev., Vol. 13, pp. 524～547 (1971).

142) R.M. Karp and R.E. Miller: Parallel Program Schemata, J. Comput. System Sci., Vol. 3, pp. 147～195 (1969).

143) J.L. Rosenfeld: A Case Study in Programming for Parallel Process, Comm. ACM, Vol. 12, pp. 645～655 (1969).

144) 吉岡: 専用プロセッサの動向，情報処理，Vol. 12, pp. 505～510 (1971).

145) J.F. Ruggiero and D.A. Coryell: An Auxiliary Processing System for Array Calculations, IBM Syst. J., No. 2, pp. 118～135 (1969).

146) C.G. Dahlquist: A Special Stability Problem for Linear Multistep Methods, BIT, Vol. 3, pp. 27～43 (1963).

147) ミクシンスキー: 演算子法，裳華房 (1964).

148) 渡部: 線形回路理論，電子回路講座第5巻，昭晃堂 (1971).

149) 渡部他: 電子回路の CAD，電子通信学会 (1973).

150) O.C. Zienkiewicz: The Finite Element Method in Structural and Continuum Mechanics, McGraw-Hill (1967).

151) R.W. Clough and E.L. Wilson: Stress Analysis of a Gravity Dam by the Finite Element Method, in Simposium on the Use of Computers in Civil Engineering (1972).

152) G. Kron: Diakoptics——Piecewise Solution of a Large-scale Systems, Electrical Journal, June (1967).

153) B.M. Irons: A Frontal Solution Program for Finite Element Analysis, Internat. J. for Numerical Methods in Engineering, Vol. 2, pp. 5～32 (1970).

154) C.W. McCormick: Application of Partially Banded Matrix Methods to Structural Analysis, in 10), pp. 155～158 (1969).

155) G.B. Dantzig: Maximization for a Linear Function of Variables Subject to Linear Inequalities, in 156), Chap. 21.

156) T.C. Koopmans (ed.): Activity Analysis of Production and Allocation, Cowles Commision Monograph No. 13, J. Wiley, New York (1951).

157) M. Simonnard: Linear Programming (trans. by W.S. Jewell), Prentice-Hall, Englewood Cliffs (1966).

158) W. Orchard-Hays: Advanced Linear-Programming Computing Techniques, McGraw-Hill, New York (1968).

159) R.S. Varga: Matrix Interative Analysis, Prentice-Hall, Englewood Cliffs, New Jersey (1962).

160) E.L. Wachspress: Iterative Solution of Elliptic Systems and Applications to the Neutron Diffusion Equations of Reactor Physics, Pre-

ntice-Hall, Englewood Cliffs, New Jersey (1966).
161) D. M. Young: Iterative Solution of Large Linear Systems, Academic Press, New York (1971).
162) B. L. Buzbee, G. H. Golub and C. W. Nielsen: On Direct Method for Solving Poisson's Equations, SIAM J. Numer. Anal., Vol. 7, pp. 627～656 (1970).
163) F. W. Dorr: The Direct Solution of the Discrete Poisson Equation on Rectangle, SIAM Rev., Vol. 12, pp. 248～263 (1970).
164) G. H. Golub: Direct Methods for Solving Elliptic Difference Equations, in Symposium on the Theory of Numerical Analysis, Lecture Notes in Mathematics 193, Springer, New York (1971).
165) R. W. Hockney: The Potential Calculation and Some Applications, pp. 135～211 in Methods in Computational Physics Vol. 9, Academic Press, New York (1970).
166) O. B. Widlund: On the Use of Fast Methods for Separable Finite Difference Equation for the Solution of General Elliptic Problems, in 12), pp. 121～131 (1972).
167) H. L. Stone: Iterative Solution of Implicit Approximations of Multidimensional Partial Differential Equations, SIAM J. Numer. Anal., Vol. 5, pp. 530～558 (1968).
168) H. G. Weinstein: Iteration Procedure for Solving Systems of Elliptic Partial Differential Equations, in 10), pp. 139～143 (1969).
169) フォーサイス・ワソー: 偏微分方程式の差分法による近似解法, 吉岡書店 (1968).
170) W. F. Tinney: Comments on Using Sparsity Techniques for Power System Problems, in 10), pp. 25～34 (1969).
171) R. Baumann: Sparseness in Power Systems Equations, in 11), pp. 105～126 (1971).
172) M. E. Churchill: A Sparse Matrix Procedure for Power Systems Analysis Programs, in 11), pp. 127～138 (1971).
173) E. C. Ogbuobiri: Sparsity Techniques in Power System Grid-Expansion Planning, in 11), pp. 219～230 (1971).
174) G. L. Guyman and I. P. King: Application of the Finite Element Method to Regional Water Transport Phenomena, in 12), pp. 115～120 (1972).

(昭和50年12月26日受付)