数　値　解　析　１４－２
（１９８５．１０．５）

パイプライン処理によるスパースな連立一次方程式の求解の高速化

田岡　久雄　　阿部　茂　　坂口　敏明
（三菱電機株式会社）

１．まえがき

近年、著しい計算機技術の進歩に伴い、アレイプロセッサやスーパーコンピュータ等のパイプライン処理を行う計算機が出現し、これらの計算機を用いた科学技術計算の高速化に関する研究が、各方面で盛んに行われている。パイプライン処理計算機は、ベクトル化を進めることにより、浮動小数点演算を高速に実行できることから、その研究の多くは、ベクトル化手法を中心に行われている。

電力系統解析の分野においても、パイプライン処理計算機を用いて、解析の中心をなすスパースな連立一次方程式の求解の高速化を図るため、種々の研究が行われてきた。これらの研究を大別すると、解析手法の面からの研究では、

(1) 従来のバンド化やブロック化の手法を用いるもの〔1〕～〔3〕
(2) 新しい解析手法を適用するもの〔4〕～〔11〕

計算機技術の面からの研究では。

(3) ハードウェアの改良によるもの〔12〕
(4) コンパイラによりベクトル化を図るもの〔13〕～〔15〕

がある。

研究が行われ始めた初期の頃は、バンド化やダイアコプティクスによるブロック対角等の手法によって、行列の一部を密化して演算のベクトル化を図る手法〔1〕～〔3〕が提案された。ところが、電力系統解析における連立一次方程式は、不規則で非常にスパースな係数行列を持つため、(1) 長いベクトル長が得られない、(2) スパース性を考慮するため二重のインデックス処理が必要となる等の理由から、思ったほどの効果が得られず、パイプライン処理計算機では電力系統解析の高速化は難しいと考えられた。

これに対し、筆者は、パイプライン処理計算機では、浮動小数点演算を高速に実行できるようになった結果、メモリアクセスや、それに伴うアドレスやパラメータの処理で計算速度が抑えられ、これらに費やされる時間が無視できなくなっていると考えた。そして、浮動小数点演算の回数を最少にすることを第一の目的に開発されてきた従来の手法〔16〕～〔21〕を今一度見直し、単にベクトル化を進めるだけでなく、メモリアクセスの回数やアドレス・パラメータの処理を減らして、パイプラインの効率を高めること、連続する演算の並列性を引き出すこと等を目的に、パイプライン処理の特性を生かした解析手法の開発を行った〔4〕～〔11〕。

筆者が研究を開始した頃は、パイプライン処理計算機は米国の数社で製品化されているのみであったが、その後、パイプライン処理計算機の有用性が認識されるにつれ、日本国内においてもパイプライン処理計算機の実用化研究が開始された。その過程で現れたものが、一つは二重インデックスの処理を高速に行うハードウェアの提案であり〔12〕、コンパイラにより自動的にベクトル化するソフトウェアツールの提案である〔13〕～〔15〕。

本論文では、スパースな連立一次方程式の求解の高速化を図るために開発した種々の新しい解析手法を、代入過程・三角化過程・オーダリング過程の３つに分けて述べると共に、電力系統解析における連立一次方程式の求解に適用して、それらの有効性を確かめる。

２．代入過程の高速化

電力系統解析に現れる連立一次方程式は、反復解法では非常に収束性が悪いことから、求解過程を三角化過程と代入過程に分けて解く直接解法が用いられる。しかも、電力系統のシミュレーションにおいては、三角化過程が２～３回行うだけでよいのに対し、代入過程は１０００回を越える計算を必要とする。

そこで、まず本章では、スパースな連立一次方程式Ａｘ＝ｂの求解中の代入過程を、パイプライン処理を行う計算機上で高速に行うための解析手法として、パイプラインの無駄を減らす新しいデータ格納方法と、演算の並列性を引き出すオーダリング手法を提案する。

2．1　新しいデータ格納方法

ガウスの消去法に代表される直接解法は、グラフ理論の上から見ると、ノードを逐次消去していく過程とみなすことができる。最初に提案する方法は、このガウスの消去法において、三角化後の値を図１に示すようなベクトルの形で格納する方法である。なお、D(l)，P(l)，Q(l)は、非零非対角要素の数だけのベクトル長で格納される。

この結果、図２に示す従来の行／列インデックス方式の場合、以下のような二重のループになっていた代入過程が、図３に示すように単一のループで実行できるようになる。

```
for i (i = 1, 2, …, n)
    ks = IROW(i)                    (1)
    ke = IROW(i+1) − 1              (2)
    for k (k = ks, …, ke)
        j = ICOL(k)                 (3)
        bj = bj − A(k)・bi
            (ただしj > iの時)       (4)
```

このため、図４で示すような、二重ループの場合に各行単位に必要だったパイプラインのセットアップ（例えば(1)・(2)式の計算）に費やす時間が不要となり、ベクトル長が格段に長くなり、パイプラインの効率が上がる。さらにi，jの値として、変数のアドレスを直接格納すれば、xとbを同一のメモリ領域に取ることで、代入過程の際のアドレスの計算を除くことができる。また、スパース性の高い連立一次方程式ではメモリ容量が節約になる。なお、ここでは対称の場合を示したが、一般の非対称係数行列の連立一次方程式についても同様のデータ格納方法を適用できる。（詳細は文献〔6〕参照）

2．2　パイプライン処理における演算の並列性

パイプライン処理計算機は、図５に示すように、乗算・加算を幾つかのステージに分けて計算し、実質的な演算速度を上げようとするものである。演算効率を上げるためには、複数の演算をオーバーラップさせながら実行する必要がある。パイプラインの各ステージで同時に行われている演算は、互いに並列に実行可能なものでなくてはならない。

今、図５に示すように、前の演算結果を以後の演算で使う場合を、演算のフィードバックと名付け、その間の演算のステップ数をそ

i	C(i)
1	1/a11
2	1/a'22
3	1/a'33
4	1/a'44
5	1/a55
6	1/a66
7	1/a77
8	1/a'88
9	1/a'99
10	1/a'1010
11	1/a'1111

l	P(l)	Q(l)	D(l)
1	2	1	a21 /a11
2	3	2	a32 /a'22
3	4	5	a45 /a55
4	4	6	a46 /a66
5	3	4	a34 /a'44
6	9	3	a93 /a'33
7	8	7	a87 /a77
8	9	8	a98 /a'88
9	10	9	a109/a'99
10	11	10	a1110 /a'1010

注）'の付いている要素は 三角化の過程で変更を受けた要素である。 P(l), Q(l)は、ノードの縮約順序を示すベクトルである。

図１　新しいデータ格納方法

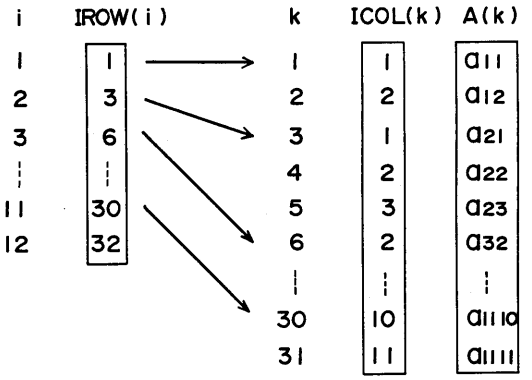


図２　行／列インデックス法による格納データ

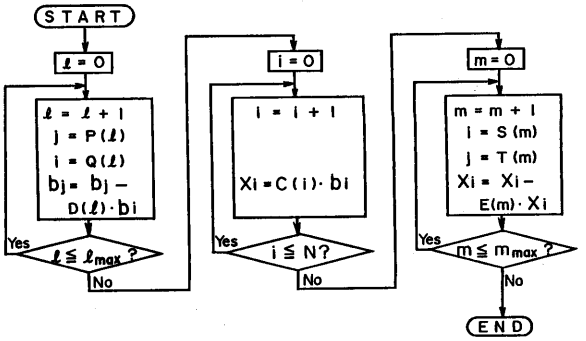


図３　提案手法における代入過程のフローチャート

のループの大きさと呼ぶことにする。この演算のフィードバックのループが大きいと、並列に実行可能な演算の数が増え、パイプラインの無駄が減り、効率を高めることができる。

連立一次方程式の求解の代入過程は、次の形の式の繰り返し計算である。

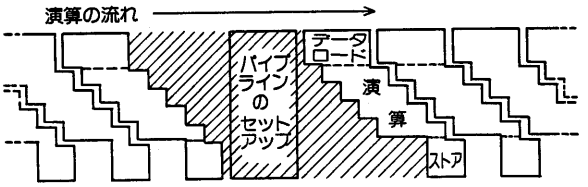
$$Z = Z - X \cdot Y \qquad (5)$$

演算結果Ｚを、次の (5)式の演算で使うことがあると、演算のフィードバックのループが小さくなり、パイプラインの効率が落ちる。ＺからＺへのフィードバックは、Ｘ・Ｙの演算が先に行われるため、パイプラインのステージ数の少ない計算機ではパイプラインの無駄は生じないので問題ないが、ＺからＸへのフィードバックは演算の無駄を生じさせる。

代入過程の前進代入の式

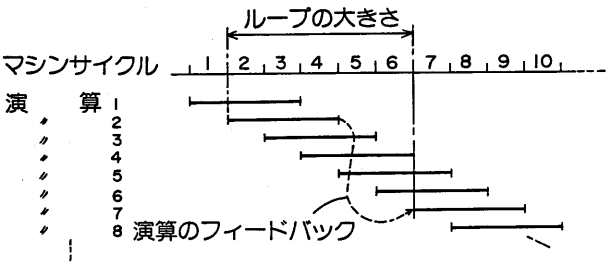
$$bj = bj - bi \cdot D(1) \qquad (6)$$

では、Ｚはｂｊに、Ｘはｂｉに相当する。ＺからＺへのフィードバックは、ｂｊの値を連続して書き換える時に発生し、これは図６(ａ)のように同一のノードに続けて縮約する場合に相当する。ＺからＸへのフィードバックは、ｂｊの値を次のｂｉとして使う時に発生し、これは図６(ｂ)のように連続するノードを逐次縮約していく場合に相当する。そこで(6) 式において、ｂｊの結果をすぐ次のｂｉとして使わないようにしなくてはならない。そのためのオーダリング手法を、以下で提案する。



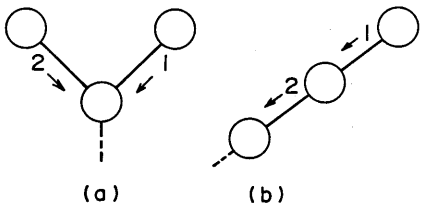
注）斜線部がパイプラインの無駄である。

図４　パイプラインのセットアップ時の無駄の発生



注）演算２の結果を演算７で使う場合を示す。

図５　パイプライン処理計算機の演算シーケンス



注）矢印の番号は、縮約の順序を示す。
以下の図も同様である。

（ａ）ＺからＺへ　　（ｂ）ＺからＸへ

図６　演算のフィードバック

２．３　演算の並列性を引き出すオーダリング手法

オーダリング手法は、処理の手間と最適化の程度から従来よりティニー・オーダー２（Tinney Order 2）、即ち、その時々の非零要素最少の行を順次選んでいく手法〔16〕が、電力系統解析において一般に使われている。この方法は、図７で示すように、系統の端のノードを順に選んでいく方法で、ツリー状の系統及びループを１つだけ含む系統の回路網方程式では、最適オーダリングを与える〔18〕。

パイプラインの無駄を減らすためには、前節の検討結果から、ｂｊの結果をすぐ次のｂｉとして使わないように、三角化を行う際、直前に選択した列の番号ｊと等しくない行ｉを順次選んでいけばよい。

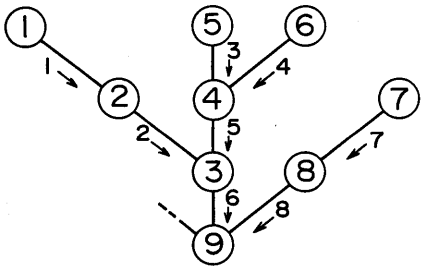


図７　ティニー・オーダー２による
オーダリング

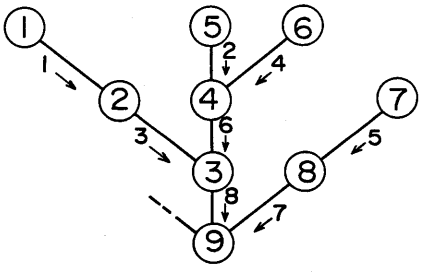


図８　新しいオーダリング手法による
オーダリング

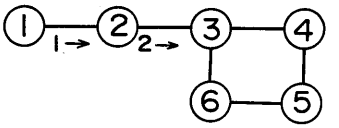


図９　ツリー状部分が１つのループ状系統例

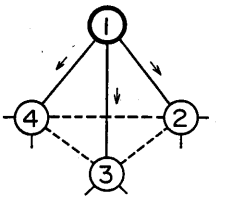


図１０　ループ状系統におけるノードの縮約

これは、図８のように、互いに隣合わないノードを順次探して縮約していくことに相当する。

そこで、この手法を前述のティニー・オーダー２と組み合せる。即ち、三角化過程において、その時の非零要素最少の行が複数ある場合、選択に自由度があることから、直前に選択した列の番号ｊと等しくない行ｉを選ぶ。ツリー状系統では、三角化過程で常に端が２つ以上存在するため、すべてのｉが直前のｊと等しくない最適オーダリングを得ることができる。同様に、ティニー・オーダー３等の他のオーダリング手法に対しても、本手法を組み合せることができる。

２．４　ループ状系統の場合

ループ状系統のオーダリングでは、行選択時の自由度がない場合が生じる。そこで、ループ状系統の場合のオーダリング手法を、以下の３つの場合に分けて検討する。

(1) ツリー状の部分が２以上ある場合

(2) ツリー状の部分が１つしかない場合

(3) ツリー状の部分が全くない場合

(1) の場合は、端が２つ以上あるため選択に自由度があり、非零要素を発生させることなく、連続する演算の並列性を引き出しながらオーダリングを実行できる。

(2) は、図９のように端が１つしかない場合である。この時は、最適オーダリングを得るアルゴリズムを優先して、ツリー状のノードを先に順次選択して、ｉが直前のｊと等しくなる場合を許すことにする。これは、最適オーダリングを崩した場合、新たな非零要素が発生して代入過程の演算の回数が増え、演算量が増加するためである。

(3) の場合は、図１０に示すようなブランチが２つ以上あるノードを選択することになる。この時の代入過程の演算は、例えば図１０のノード１の場合、

$$b2 = b2 - b1 \cdot a21 / a11 \quad (7)$$

$$b3 = b3 - b1 \cdot a31 / a11 \quad (8)$$

$$b4 = b4 - b1 \cdot a41 / a11 \quad (9)$$

の３つである。これらの演算順序は任意であり、非零要素の発生には無関係である。そこで、次のステップでノード４が選択される場合、先に、ノード４の値を使う(9) 式の演算を実行する。すると、次のステップで選択に自由度がないにもかかわらず、連続する演算に並列性を持たせることができる。即ち、直前に選択された列番号ｊと等しくない行番号ｉを選択することは、裏返せば次に選択する行番号ｉと等しい列番号ｊの演算を先に行うことである。つまり、列番号の選択に対しても、最適オーダリングを得るためのアルゴリズムを適用すればよい。なお、直前に選択されたノードがツリー状の部分であった場合は、(2) の場合と同様に、最適オーダリングを得るアルゴリズムを優先しなければならない。これは、(2) から変化した系統状態である。

以上のことから、ループ状系統のオーダリングでは、ツリー状の部分が１つある状態の時のみ、最適オーダリングを得るためのアルゴリズムを優先する必要があるが、その他の場合は、列番号の選択に対しても最適オーダリングを得るためのアルゴリズムを適用することにより、最適オーダリングを得ながら、かつ連続する演算の並列性を引き出すオーダリングを実行することが可能である。

なお後退代入では、ｉとｊが入れ替るが、逆の順序で式を計算するため、上述のオーダリングで、必然的に演算のフィードバックのループを大きくできる。

2．5　多段・複数のパイプラインがある場合

段数が多いパイプライン処理計算機では、連続する３つ以上の演算についても考慮する必要がある。この場合には、パイプラインの段数が増すにつれ、以下のアルゴリズムを適用していく。

(1) ｊについても、直前で選択されたｊと等しくないものを選ぶ。

(2) ｉについて、直前で選択されたｊだけでなく、２ステップ以上前で選択されたｊとも等しくないものを選ぶ。

(3) ｊについても、(2)と同様の方法で選択する。

さらに、複数の乗算器・加算器を持つパイプライン処理計算機では、同時に２つ以上の行を扱うことができるため、ｉ・ｊの番号が互いに重ならないように演算を選択していく。

3．三角化過程の高速化

本章では、三角化過程における浮動小数点演算部分を、オーダリング・インデックス等の処理と分離してベクトル化を行い、三角化過程の高速化を図る手法を提案する。

3．1　三段解法

三角化過程を分析してみると、方程式の係数行列の値を使った浮動小数点演算の占める割合は約２割であり、残りはオーダリングやインデックスの処理で占められている。そこで、三角化過程の高速化を図るため、三角化過程における浮動小数点演算部分を、オーダリング・インデックス等の処理から分離し、連立一次方程式の求解を、(1) オーダリング・インデックス処理過程、(2) 三角化過程、(3) 代入過程の３段階に分けて解くことにする。

電力系統解析の対象は、同一系統あるいは系統の一部の値を変更した場合のケーススタディが多く、系統の構成はほとんど変更がないのが特徴であることから、(1) の過程をあらかじめ行っておけば、毎回の計算は(2),(3) の過程のみを行えばよく、種々のケーススタディを高速に処理することが可能となる。

3．2　テーブルの作成

三角化過程における浮動小数点演算は、表１に示す４種類である。そこで、オーダリング・インデックス処理過程において、表１に挙げた〔１〕～〔４〕の４種類の浮動小数点演算の計算手順を示すテーブルを作成する。表に示すように、ＩＡには演算の種類を示す１～４の値を、ＩＢには入力データのアドレスを、ＩＣには出力データのアドレスを格納する。〔２〕・〔４〕の演算には直前のＣの値が、〔３〕の演算では直前のＤの値が必要であるが、これらの値はレジスタ経由で使うことにする。

すると、三角化過程において、オーダリング・インデックス処理過程で作成したテーブルによって、４種類の演算が図１１に示したフローチャートに従って、逐次選択されながら単一ループで実行されることになり、ベクトル化が容易になる。

表１　三角化過程用テーブルの内容

	演算内容	演算に必要なマシンサイクル数	ＩＡ（値）	ＩＢ（アドレス）	ＩＣ（アドレス）
〔１〕	C(i)＝１／a ii	a (28)	1	a ii	C(i)
〔２〕	D(l)＝C(i)・a ji	b (3)	2	a ji	D(l)
〔３〕	a jk＝a jk－D(l)・a ik	c (5)	3	a ik	a jk
〔４〕	E(m)＝C(i)・a ik	b (3)	4	a ik	E(m)

（注）　( )内の値は、ＡＰ－１２０Ｂの場合を示している。

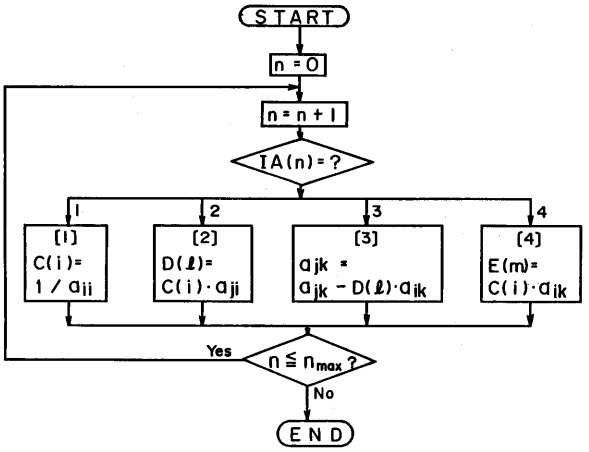
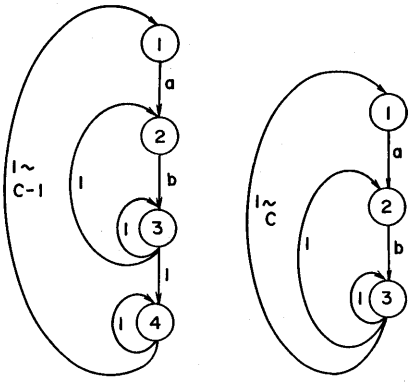


図１１　提案手法における三角化過程のフローチャート



（ａ）非対称の場合　（ｂ）対称の場合

図１２　三角化過程の各演算間の必要マシンサイクル

図１２には、三角化過程の各演算間で費やされるマシンサイクル数を示す。ａ～ｃは各々〔１〕～〔３〕の演算実行に費やされるマシンサイクル数であり、１はオーバーラップして連続演算が可能なことを示している。図に示すように、〔３〕の結果ａjj（ｋ＝ｊの場合）を次の〔１〕で使う時、非対称の場合〔４〕→〔１〕でｃ－１マシンサイクル、対称の場合〔３〕→〔１〕でｃマシンサイクル費やす。これを連続演算可能にするためには、第２章で提案したオーダリング手法を、オーダリング・インデックス処理過程で適用すればよい。

オーダリングを先に行う事はよく使われる手段であるが、本手法は、オーダリングと共に、三角化過程の演算手順をあらかじめテーブルの形で格納することで、三角化過程の浮動小数点演算のベクトル化を図っている。なお、計算精度を改善するためのピボッティングは、方程式の係数行列の値の大小を比較する必要があるため本手法では行うことができないが、電力系統解析ではピボッティングを省略できる場合が多い。

４．オーダリング過程の高速化

スパース性を考慮したために必要となるオーダリング・インデックス等の処理過程は、整数演算とメモリアクセスがほとんどを占めており、浮動小数点演算の高速化を目的としたパイプライン処理計算機では、高速化が難しい。

本章では、適用アルゴリズムの改良により高速化が可能なオーダリング過程の高速化を図るため、スパース性を利用してクイックソートのキーを固定することにより、高速にソーティングを行うことのできる改良クイックソートと、それを適用したオーダリングの手法を提案する。

(1) [2, 1, 3, 4, 2, 2, 1, 3, 1, 1]　キー＝２
(2) [2, 1, 1, 4, 2, 2, 1, 3, 1, 3]　キー＝２
(3) [2, 1, 1, 1, 2, 2, 1, 3, 4, 3]　キー＝２
(4) [2, 1, 1, 1, 1, 2, 2, 3, 4, 3]　キー＝２
(5) [2, 1, 1, 1, 1] 2 [2, 3, 4, 3]　キー＝２
(6) [1, 1, 1, 1] 2, 2 [2, 3, 4, 3]　キー＝１
(7) [1, 1, 1] 1, 2, 2 [2, 3, 4, 3]　キー＝１
(8) [1, 1] 1, 1, 2, 2 [2, 3, 4, 3]　キー＝１
(9) 1, 1, 1, 1, 2, 2 [2, 3, 4, 3]　キー＝２
(10) 1, 1, 1, 1, 2, 2, 2 [3, 4, 3]　キー＝３
(11) 1, 1, 1, 1, 2, 2, 2 [3, 3, 4]　キー＝３
(12) 1, 1, 1, 1, 2, 2, 2, 3, 3, 4

注）↑の要素を交換する

図１３　クイックソートによるオーダリングの手順

4．1　改良クイックソート

電力系統解析に現れるスパースな連立一次方程式の求解過程のオーダリングは、以下のような特徴を持つ配列問題である。

(1) 同一要素の多い配列の並べ換え問題である。

(2) 要素の値は１～１０程度の範囲を取るものが多い。

従来より、オーダリングには、ソーティングのアルゴリズムが使われている。ソーティングのアルゴリズム〔22〕～〔25〕では、配列要素がすべて互いに異なる場合を想定しているため、このような同一要素が多く、しかも１～１０程度の狭い範囲の値をとる配列のソーティングでは、同じ値の要素の交換を不必要に実行し、必ずしも効率よく並び換えが行われるとは限らない。例えば、図１３のクイックソートによるオーダリングの例では、(６)～(８)のステップで、無駄なソーティングを繰り返し実行している。

ここで提案する手法は、クイックソートに制限を加えて、オーダリングを高速に実行するソーティング手法である。即ち、

(1) キーを１から順に選ぶ。

(2) 配列の左側からキーより大きいものを、右側からキーより小さいか等しいものを探索して互いに交換していき、キーより左側にある要素は、すべてキーより小さいか等しいものが、キーより右側にある要素は、すべてキーより大きいものがくるように並び換える。

すると、図１４に示すような手順でソーティングが行われる結果、左側から順次ソーティングが終了し、左側からの探索が終了した時点でソーティングが終了することになる。また、

(1) 分割されないため、スタックが不要である。

(2) 先頭から順番にソーティングが終了するため、左側からのサーチは一度でよい。

ことから、従来のクイックソートに比べて格段の高速化が可能となる。

(1) [2, 1, 3, 4, 2, 2, 1, 3, 1, 1]　キー＝1

(2) [1, 1, 3, 4, 2, 2, 1, 3, 1, 2]　キー＝1

(3) [1, 1, 1, 4, 2, 2, 1, 3, 3, 2]　キー＝1

(4) [1, 1, 1, 1, 2, 2, 4, 3, 3, 2]　キー＝1→2

(5) [1, 1, 1, 1, 2, 2, 4, 3, 3, 2]　キー＝2

(6) [1, 1, 1, 1, 2, 2, 2, 3, 3, 4]　キー＝2→3

(7) [1, 1, 1, 1, 2, 2, 2, 3, 3, 4]　キー＝3→4

注)　↑の要素を交換する

図１４　改良クイックソートによるオーダリングの手順

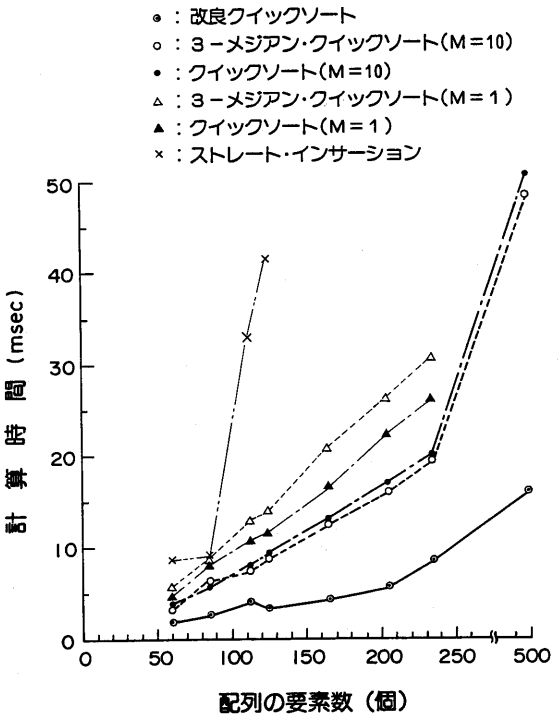


図１５　各種ソーティング手法の比較

この改良クイックソートを、オーダリングに適用した時の、ソーティング処理に費やす時間を、他のソーティング手法と比較したのが、図１５である。Mは、ストレート・インサーション法に切り換える際の配列の長さである。図より、従来の手法の中で最も速いとされているクイックソートと比べて、約２倍の高速化が達成されていることがわかる。

4．2　オーダリングへの適用

オーダリングの代表的なものとして、ティニー・オーダー１～３の３つの手法がある。

ティニー・オーダー１は、非零要素の少ない順に行番号を最初に並べておくオーダリング手法であるため、改良クイックソートをそのまま適用すればよい。

ティニー・オーダー２及びティニー・オーダー３は、ダイナミックにオーダリングを行う手法である。ティニー・オーダー２は、各ステップの行番号選択時に、その時の非零要素の最も少ないものを、ティニー・オーダー３は、その時の非零要素の発生の最も少ないものを選ぶ手法である。そこで、非零要素の数を示す配列の並び換え、あるいは非零要素の発生する数を示す配列の並び換えに、それぞれ改良クイックソートを適用する。即ち、まず、

(1) 初期の配列に対し、改良クイックソートを適用して要素の値の小さい順に並び換える。その際、各要素数の先頭の位置をポインターとして持っておく。

途中で配列の要素の値が変化した時は、

(2) 値が減少した時は、図１６(ａ)に示すように、減少する前の要素の値のポインターの示す要素と減少した要素を交換し、ポインターを１だけ増す。

(3) 値が増加した時は、図１６(ｂ)に示すように、増加した後の要素の値のポインターの示す位置より１つ前の要素と増加した要素を交換し、ポインターを１だけ減らす。

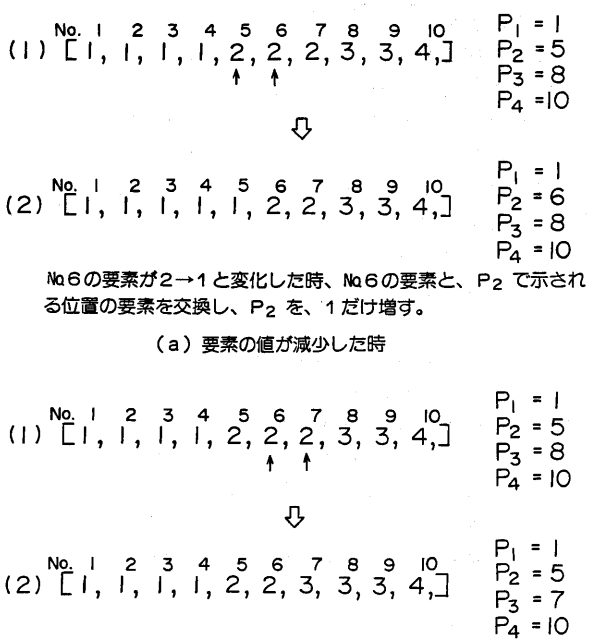


図１６　オーダリングの途中における処理手順

この方法を用いれば、オーダリングの各ステップで毎回ソーティングを行う必要がなくなり、行選択時の手間を最小限に抑えることができる。

5．計算時間の比較

電力系統の回路網方程式の求解に対して、本章で提案した新しい求解法を適用し、従来の求解法による場合と代入過程の計算時間の比較を行った結果が、図１７、図１８である。

図１７は代入過程、図１８は三角化過程の場合を示す。共に(ａ)は、アレイプロセッサＡＰ－１２０Ｂ上で、提案手法による場合と従来の行／列インデックス法をそのまま適用した場合の、計算時間を比較したものである。(ｂ)は、提案手法のＡＰ－１２０Ｂ上での従来の手法に対するスピードアップ率、及び提案手法をＡＰ－１２０Ｂ上で実行した場合をスーパーミニコンピュータＶＡＸ－１１／７８０上で実行した場合と比べた時のスピードアップ率を示す。なお、１１、６３、８３、１０３母線系統はツリー状、１４、３０、５７、１１８母線系統はループ状、４３、２５０母線系統はツリーとループの混在系統であり、方程式は複素対称行列を係数行列に持つ。

図より、代入過程において、アレイプロセッサＡＰ－１２０Ｂ上で、従来の手法と比較して、平均して約３倍の高速化が達成されていることがわかる。ＡＰ－１２０Ｂの最高演算速度が１２MFLOPSであるのに対し、新しい求解法では８MFLOPSの演算速度が得られており、パイプラインの効率化が行われている。

三角化過程においては、アレイプロセッサＡＰ－１２０Ｂ上で従来の手法と比較して、小さな系統で４～５倍、大きな系統では２０～５０倍の大幅な高速化が達成されており、従来の手法に比べて、浮動

小数点演算部分のみを抜き出してベクトル化を図った本手法の効果が、著しいことがわかる。

スーパーミニコンピュータＶＡＸ－１１／７８０上で実行した場合と比較すると、代入過程、三角化過程共に、パイプライン処理の効果により２０倍程度のスピードアップが達成されていることがわかる。

図１９は、電力系統のシミュレーションにおける回路網方程式の求解に対する提案手法の効果を示したものである。オーダリング・インデックス処理の過程は、多量のメモリアクセスを伴うことから、改良クイックソートの適用によるオーダリング手法の高速化の効果は少ないが、その他の過程の高速化と合せて、全体で約５０％のスピードアップが達成されている。

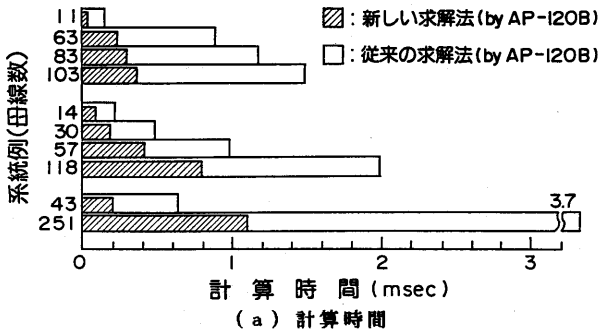
電力系統のシミュレーション全体では、図２０に示すように、三角化過程は３回で済むのに対し、代入過程が１０００回を超える計算が必要であることから、第２章で提案したデータ格納方法とオーダリング手法の効果が著しいと言える。また、第３章で提案した三角化過程の演算手順のテーブル化による浮動小数点演算部分の分離により、オーダリング・インデックス処理の過程を一回で済ますことができるため、大幅な時間短縮が達成されている。以上の結果、シミュレーションにおける連立一次方程式求解部分全体でみると、スピードアップ率は９８％と非常に著しい。

６．あとがき

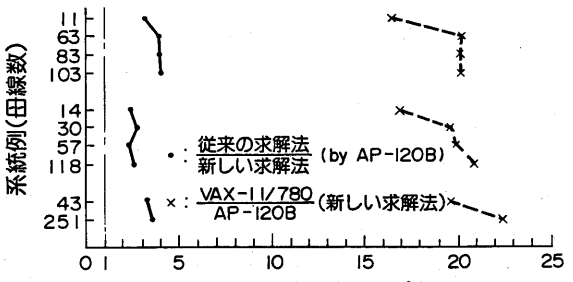
本論文では、スパースな連立一次方程式の求解を、スーパーコンピュータやアレイプロセッサ等のパイプライン処理を行う計算機上で高速に行うための、新しい解析手法について提案した。即ち、

(1) データ・アドレスのベクトル化による新しいデータ格納方法
(2) 連続する演算の並列性を引き出すオーダリング手法
(3) 三角化過程の演算手順のテーブル化による、三角化過程のベクトル化手法
(4) クイックソートのキーを固定して高速化を図った改良クイックソート
(5) 改良クイックソートを適用したオーダリング手法

である。

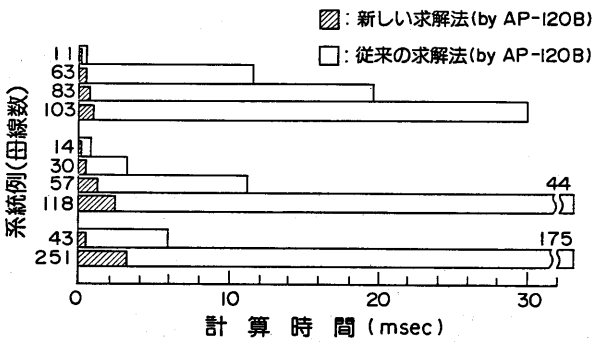


（ａ）計算時間

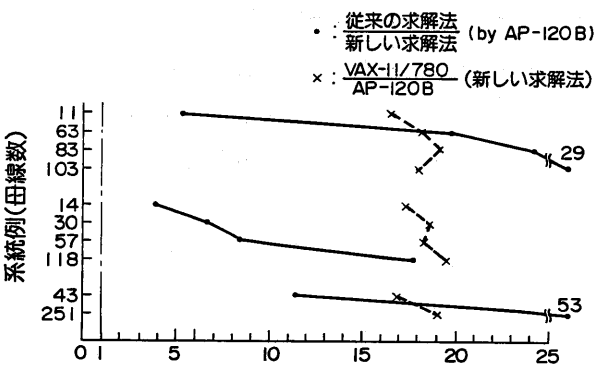


（ｂ）スピードアップ率

図１７　代入過程の計算時間の比較

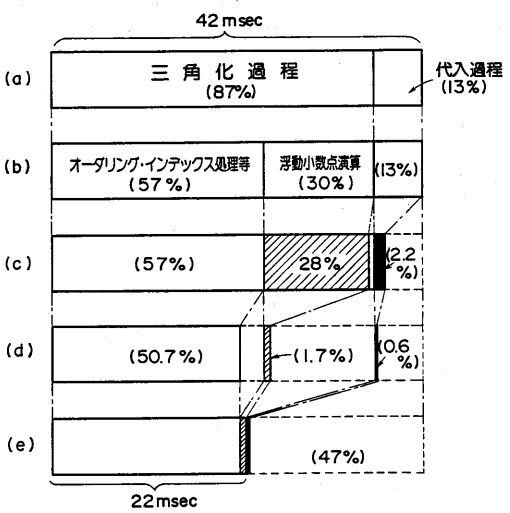


（ａ）計算時間



（ｂ）スピードアップ率

図１８　三角化過程の計算時間の比較



▨及び■ は AP-120B, 他は VAX-11/780を使用している。

(a): 従来の求解法の計算時間

(b): 三角化過程における浮動小数点演算の比率

(c): AP-120Bの数値計算ライブラリによる時間短縮の効果

(d): 新しい求解法による時間短縮の効果

(e): 新しい求解法の計算時間

図１９　一回の連立一次方程式求解における計算時間の短縮の様子

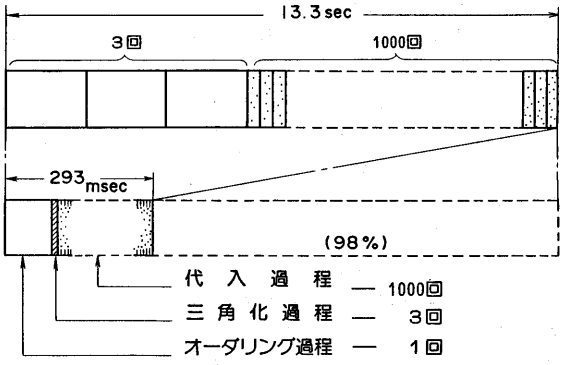


図２０　電力系統のシミュレーション中の連立一次方程式求解部分における計算時間の短縮の様子

また、これらの手法を電力系統解析の回路網方程式の求解に適用して、従来の手法による場合と比較し、提案手法の有効性を確かめたところ、その効果は非常に著しいことがわかった。

本手法は、電力系統解析のみならず、構造解析や物理計算等、スパース性を持つ、他の分野の各種の問題にも適用可能であり、スーパーコンピュータやアレイプロセッサ等のパイプライン処理計算機上で用いることにより、大幅な高速化が期待できる。

参考文献

〔1〕 "Exploring Applications of Parallel Processing to Power System Dynamic Analysis Problems," EPRI Special Report, EL-566-SR (1977)

〔2〕 J.P. Hulskamp, S.M. Chan & J.F. Fazio, IEEE Trans. PAS, Vol. PAS-101 (1982)

〔3〕 小国・後・西方・長堀　情報処理学会論文誌　25巻　5号　(昭59-9)

〔4〕 田岡・阿部　昭59電気学会全大　No. 868 (昭59-3)

〔5〕 田岡・阿部　電気学会情報処理研究会 IP-84-5　(昭59-5)

〔6〕 田岡・阿部　電気学会論文誌B　105巻 5号　(昭60-5)

〔7〕 田岡・阿部　情報処理学会第30回全大 No. 6D-8　(昭60-3)

〔8〕 H. Taoka & S. Abe, Proc. 1985 PICA Conference, (May 1985)

〔9〕 田岡・阿部　電気学会電力技術研究会 PE-84-94　(昭59-7)

〔10〕 田岡・阿部・坂口　電気学会情報処理研究会　IP-85-7　(昭60-2)

〔11〕 田岡・阿部・坂口　電気学会電力技術研究会　PE-85-46　(昭60-7)

〔12〕 阿部・高藤・坂東・平沢・加藤　情報処理学会論文誌　25巻　4号　(昭59-7)

〔13〕 稲上・村山・長島　情報処理学会第28回全大　No. 1C-2　(昭59-3)

〔14〕 田中・安村・金田　情報処理学会第28回全大　No. 1C-3　(昭59-3)

〔15〕 佐川・鈴木・山田・中田　情報処理学会第28回全大　No. 1C-1　(昭59-3)

〔16〕 W.F. Tinney & J.W. Walker, Proc. IEEE, Vol. 55, No. 11 (1967)

〔17〕 W.F. Tinney & C.E. Hart, IEEE Trans. PAS, Vol. PAS-86, No. 11 (1967)

〔18〕 E.C. Ogbuobiri, W.F. Tinney & J.W. Walker, IEEE Trans. PAS, Vol. PAS-89, No. 1 (1970)

〔19〕 B. Stott & E. Hobson, Proc. IEE, Vol. 118, No. 1 (1971)

〔20〕 I.S. Duff, Proc. IEEE, Vol. 65, No. 4 (Apr. 1977)

〔21〕 大附、川北　情報処理　17巻　1号　(昭51)

〔22〕 C.A.R. Hoare: "Quicksort," Computer J., Vol. 5, No. 4 (Apr. 1962)

〔23〕 D.E. Knuth: "The Art of Computer Programming, Volume 3/ Sorting and Searching," Addison-Wesley Publishing Company (1975)

〔24〕 C.R. Cook & D.J. Kim, Commun. ACM, Vol. 23, No. 11, pp. 620-624 (Nov. 1980)

〔25〕 R.L. Wainwright, Commun. ACM, Vol. 28, No. 4, pp. 396-402 (Apr. 1985)