スーパコンピュータS－810向け構造解析プログラム ISASⅡ／HAP

原野紳一郎，小国力（㈱日立製作所）
小割健一（日立コンピュータコンサルタント㈱），坂本隆俊（㈱日本ビジネスコンサルタント）

1．まえがき

CRAY－1をはじめとするスーパコンピュータが世界で稼動しはじめたのが1976年であり、日本でも1980年から稼動している。世界でのスーパコンピュータの設置状況を見ると，1976年が3台，1980年が21台，1984年が120台と指数関数的に増大の一途をたどっている。しかしながら、ソフトウェアの面から見ると、まだまだごく一部の限られた分野だけに使用されることが多い。一つにはスーパコンピュータが科学技術計算の特殊分野で最大限に効果を発揮するように設計されて来た（あるいはそういう設計思想を持つものがスーパコンピュータともいえる）からである。他方，汎用機の性能も飛躍的に向上して来てはいるが，その向上分だけではカバーしきれない膨大な計算を瞬時に処理したいという長年の夢をかなえるものとしてスーパコンピュータは登場した。その意味ではスーパコンピュータは十分に責務を果たしていると思われるが、最近の世の中の状況は、それ以上の，もっと広範囲な適用を、スーパコンピュータを含めたこれからの計算機に求めている。

世の中の流れとしても、高度成長期に見られた重厚長大の思想から一転して、安定成長期の軽薄短小の世界へと移行しつつある。低成長経済下では企業間競争が熾烈になり、開発サイクルの短い製品をいかにタイムリーに世に出すかが、企業生命を左右しかねない状況にある。経済効率を上げるため，製品はより軽量に、素材の改良も行われていく。その一方では、今日のように社会に対する責任や公共性を問われている航空機，自動車，鉄道などの輸送機関は，その安全性確保のためには最大限の努力が求められている。これからの計算機に求められる期待の一つは，この経済性と安全性確保のための技術的進歩であろう。計算機に寄せる第二の期待は，従来の演算中心のコンピュータから、経験、知識，推論，判断を重視した分野への応用である。人工知能を実用化するための高性能推論型データベースマシンが必要であり、実用化に入った段階で、現在のスーパコンピュータに見られる技術に類したものが必要となろう。

スーパコンピュータは何をなし得るか。現状では，まだまだスーパコンピュータに対する期待と現実のギャップは大きいと思う。そのギャップを埋める努力が精力的に行われているが，いろいろと問題はある。例えば、ソフトウェア自身をできるだけスーパコンピュータ向きに作りかえる必要があったり，専用プログラムでは効果が出ても、汎用プログラムに対してはスーパコンピュータの効果が期待できない場合もよくあるからである。

本論文では汎用構造解析プログラムISASⅡ／HAPをスーパコンピュータ上で有効に稼動させるためには，連立一次方程式の解法に使用するアルゴリズムや演算のタイプに何を選ぶか，並列演算器を有効に利用するにはどうしたらよいか，又何にも増してI／Oネックの問題をどのように解消するかについて報告する。スーパコンピュータにとって内積型演算は最良の演算タイプではないが，内積型演算を主要部に持つスカイライン法は、その演算のDOループ長が比較的長

いためにベクトル化による加速率が割合大きく，並列演算器を有効に使うペアドスカイライン法まで拡張すれば十分効果が得られる。又，半導体拡張記憶装置を利用するようにI/O処理を変更すれば，経過時間はほとんどCPU時間と同程度になり，18000自由度の静的構造解析がS－810で5～6分程度で解く見通しがついた。

以下、2章では有限要素法と連立一次方程式解法の発展の経過を示し、第3章ではスカイライン法の概略を述べ，第4章でS－810の内積演算の性能を評価し，第5章はスカイライン法をS－810に適用した結果を報告する。第6章ではS－810を有効に利用する方法を示し，第7章と第8章でペアドスカイライン法及び拡張記憶を適用する場合の性能を予測し評価する。

2．有限要素法と連立一次方程式解法

航空機の設計・開発を端緒として発展した有限要素法は電子計算機の飛躍的発展と共に1960年代後半から1970年代前半にかけて、汎用構造解析ソフトウェア群を作り出した。その発展には差分法に見られるような領域のメッシュの規則性等の制限が緩和され、任意形状の連続体が取扱えるようになったことも大きく寄与している。有限要素法は基礎となる有限要素の研究に多大の努力が払われ、何十，何百という汎用，専用ソフトウェアとなって開花した。同時に数値解析の手法も多く研究され、有限要素法による解析に強力な手段を提供して来た。

構造解析は有限要素法の適用に最も成功した分野である。構造解析に有限要素法を適用した場合の数値解析手法についてとり上げる。図1に構造解析での有限要素解析フローを示す。

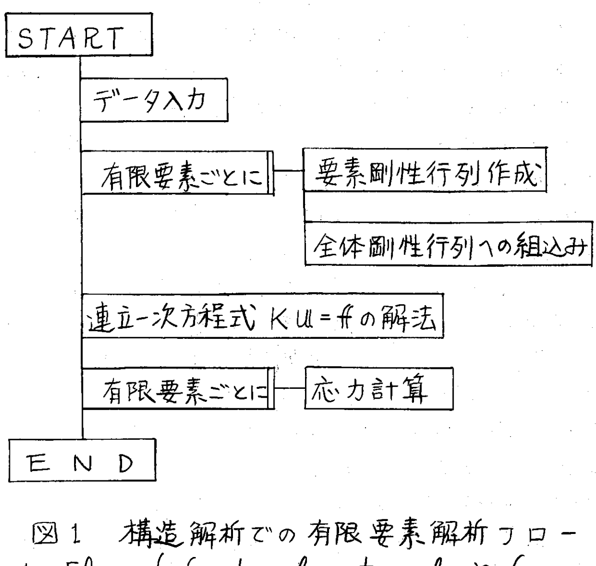

図1　構造解析での有限要素解析フロー
Fig.1　Flow of finite element analysis for structural analysis problem.

処理時間の面から見ると連立一次方程式を解く部分の比重が最も大きい。一般に構造解析に現われる演算は行列の積和と連立一次方程式を解くという類の演算に大別される。固有値計算にしても数値積分にしてもこの域を出ない。この中でも連立一次方程式解法は演算時間としても大半を占めることが多い。

連立一次方程式を解く方法は直接法と反復法に大別されるが，構造解析の分野では直接法が圧倒的に多い。直接法によるアルゴリズムは１８２６年に発表されたGaussの消去法を基礎としている。これらを発展させた形でCholeski法（１９１６年），Crout法（１９４１年）などが発表され，１９６０年代になって一斉に開発された構造解析プログラムではバンドマトリックス法，ユニット分割法（ＦＲＡＮ），ハイパーマトリックス法（ＡＳＫＡ）などが取り入れられた。又，MeloshとBamfordはウェーブフロント法を提案した。バンド内のゼロ要素を除外し，フロント行列と呼ばれる中間行列をメモリ上に保持するストア型の解法で、ＭＳＣ／ＮＡＳＴＲＡＮ，ＡＮＳＹＳ等に取り入れられて、大きな効果を上げている。一方，カリフォルニア大を中心としたＳＡＰファミリーのプログラムは，ブロック消去法（ＳＡＰ４，ＳＡＰ５）からスカイライン法（ＳＡＰ６／７，ＡＤＩＮＡ，ＩＳＡＳⅡ）の内積型の解法に移行している。現在，構造解析における直接法で行列三角分解による連立一次方程式解法の中心はウェーブフロント法とスカイライン法である。アルゴリズの特性に帰因して演算タイプがストア型と内積型に分かれることや、これからのスーパコンピュータ，技術発展の方向がどちらに味方していくのかも大いに興味の持たれる所である。

連立一次方程式といっても分野によって，又解く問題によって種々の特性を持つため，それに応じた解法が必要となって来る。構造解析に現われる行列の特徴の一つは、大規模なスパース性を持つことである。有限要素の精度によってメッシュ分割する細かさも異なって来るが、精度を上げるためにはどうしてもメッシュをある程度細かくする必要がある。それにつれて行列は大型になる。エンジニアリングセンスを駆使して、むやみにメッシュを切るという安易で誤まった方法はとられないとしても，やはり行列は大型になることは否めない。

第二には，正定値行列が大半を占めることである。これはエネルギー

$$\frac{1}{2}(u, Ku) > 0 \qquad u \neq 0 \tag{1}$$

が消失しない限り保証される。さらに，対称行列であれば，ピボットなしの行列三角分解法が適用できる。このあたりが線形計画法などに現われる行列と大きく異なる所である。線形計画法では対称性や正定値性はほとんど期待できない。構造解析を含む有限要素法がコンピュータの利用によって大きな成功を収めた理由も、一つはこのあたりにある。

３．スカイライン法

ここでは、アレイプロセッサ用総合構造解析システムＩＳＡＳⅡ／ＨＡＰ（<u>I</u>ntegrated <u>S</u>tructure <u>A</u>nalysis <u>S</u>ystem <u>II</u>／<u>H</u>igh-speed <u>A</u>rray <u>P</u>rocessor）の母体となるＩＳＡＳⅡのスカイライン法について概要を説明する。

ＩＳＡＳⅡ／ＨＡＰはスーパコンピュータＳ－８１０向けに開発を予定した汎用構造解析プログラムであるが、その基本技術はＩＳＡＳⅡから受け継いでいる。そのＩＳＡＳⅡはＮＡＳＴＲＡＮレベル１５.５.１を母体としたＩＳＡＳプログラ

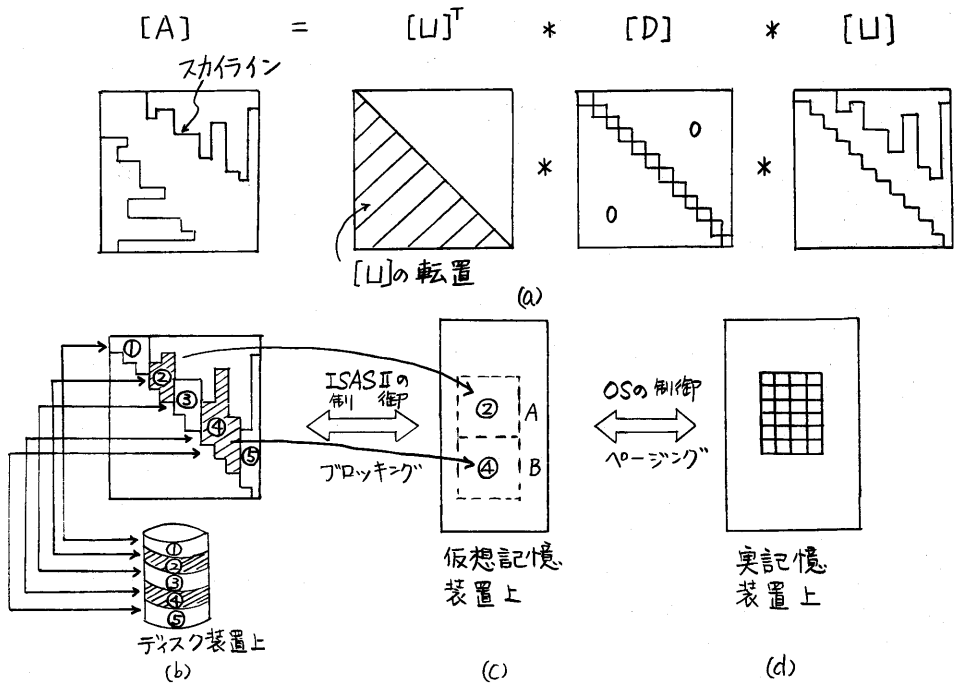

図2　スカイライン法による行列の三角分解

Fig. 2　Matrix Triangular Decomposition by Skyline method

ムから出発しており、ＩＳＡＳで使用していた「アクティブカラムを考慮したバンドマトリックス法」をブロック処理を伴なうスカイライン法に変更して高速化をはかったものである。

ＩＳＡＳⅡで採用しているスカイライン法を図２に示す。スカイライン法は上三角行列［Ｕ］を基本とする。対角行列［Ｄ］は，［Ｕ］の対角項上に作成すると、［Ｄ］用のファイルを新しく設ける必要がない。アルゴリズムは次のようになる。

$$u^*_{ij} = a_{ij} - \sum_{k=1}^{i-1} u_{ki}\, u^*_{kj} \qquad (i=2,\cdots,j-1) \qquad (2)$$

$$u_{ij} = u^*_{ij} / d_{ii} \qquad (i=1,\cdots,j-1) \qquad (3)$$

$$d_{jj} = a_{jj} - \sum_{k=1}^{j-1} u^*_{kj}\, u_{kj} \qquad (j=2,\cdots,n) \qquad (4)$$

但し　$u^*_{1j} = a_{1j}$, $d_{11} = a_{11}$　(5)

メモリ上にすべて行列成分を格納できない場合を考慮して，行列を図２（ｂ）のようにブロック化する。一つのブロックは図２（ｃ）の仮想記憶装置上のエリアＡ（又はＢ）に納まる大きさとする。ブロックの大きさが決まると，各ブロック間の関連が調べられる。その行列のブロック時性の例を表１に示す。メモリ上の一次元配列をＸ，Ｘ上でのエリアＡ，Ｂの先頭のアドレスをＩＡ，ＩＢ，対角

表1　行列のブロック特性
Table 1　Property of matrix blocks

	ブロック番号				対角成分のポインタ					
	C(1)	C(2)	C(3)	C(4)	L(J)					
ブロック1	1	1	1	6	1	3	6	8	12	15
ブロック2	2	1	7	9	4	7	14			
ブロック3	3	2	10	11	4	8				
ブロック4	4	1	12	12	11					

C(1)　：値を完成する必要のあるブロック番号
C(2)　：C(1)のブロックと相関する最小ブロック番号
C(3)　：C(1)のブロック中の最小列番号
C(4)　：C(1)のブロック中の最大列番号
L(J)　：C(1)のブロック中の第J列の対角成分のポインタ

成分を格納するエリアの先頭アドレスをIDとする。又，行列の特性に関する記号については表1のものを使用する。行列の元数をN，ブロック数をNBとする。
ブロックを考慮したスカイライン法の処理は次のようになる。

$$P=1, \cdots NB \quad (P：現在のブロック番号) \tag{6}$$

に対し，P≠1のとき，エリアAとBのアドレス及び行列特性データを交換する。

$$IA \rightleftarrows IB \quad , CA(\ell) \rightleftarrows CB(\ell) \quad (\ell=1\sim4) \tag{7}$$

グループPを作成するのに必要な最小ブロック番号をQとすると

$$Q = CA(2) \tag{8}$$

P≠QのときQ＝P－1ならば，ブロックQは既にエリアBにあり、Q≠P－1ならば、ブロックQの行列特性をファイルから読み込み，CB(ℓ) (ℓ=1〜4)にセットする。ブロックPとQの間の影響を(2)式から計算してブロックPに加え込む。これがQ＝P－1になるまでQを1ずつ増加して同様の処理をする。Q＝Pのときはエリアにある自分との相関を計算する。ブロックPについて以上が完了したらディスク上にその結果を書き出す。以上を各ブロック（1≦P≦NB）ごとに行うと、ディスク上に上三角行列が列方向に作成される。
相関を計算する部分で主要な部分は（2）式であり、

$$X(IJ) = X(IJ) - \sum_{K=NS}^{NE} X(IB+K+1) * X(IA+K+IC-1) \tag{9}$$

という形の内積計算をすることになる。

4、S－810での内積演算性能

スカイライン法はその主要部を（9）式の内積演算で構成している。3種類のコーディングタイプ，即ち，(a)内積の配列中のインデックスがDOループ変数であるもの（完全内積型），(b)インデックスが算術式であるもの（インデックス明示型，(c)完全内積型ルーチンをコールするもの（サブルーチン内積型）についてS－810での測定結果を図3に示す。測定はスカラーモードとベクトルモード

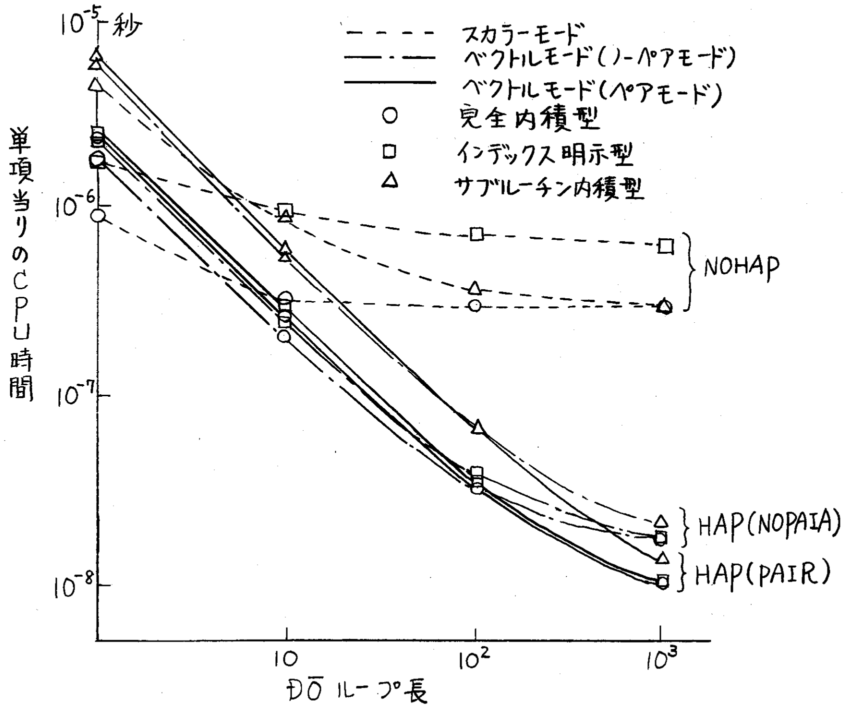

図3　S－810での内積計算CPU時間比較

Fig.3　CPU time versus DO loop length in various types of inner product operations (S-810)

表2　内積計算でのHAP加速率(完全内積型)

Table 2　HAP-Acceleration ratio of inner product operations

DOループ長	1	3	5	7	10	30	50	70	100	300	500	700	1000	3000
ノーペアモード	0.5	0.8	1.1	1.4	1.8	4.5	6.1	7.7	9.4	14.0	16.5	17.0	17.7	18.6
ペアモード	0.4	0.6	0.8	1.1	1.4	3.5	5.6	7.3	9.7	20.1	25.9	27.2	30.9	35.4

それぞれ行った。ベクトルモードは並列演算器を強制的に使用するペアモードとコーディングに応じて並列演算器を使用するノーペアモードの両方でテストした。

まず，第一にわかることは，DOループ長によって成分1個を計算するのにかかるCPU時間が大きく変化することである。表2にノーペアモードとペアモードでのHAP加速率(＝スカラーモードCPU時間÷ベクトルモードCPU時間)を示す。

加速率が平坦になるDOループ長はノーペアモードが500付近，ペアモードは3000以上である。DOループ長が100以下ではノーペアモードの方が加速率は高く，100を越えるとペアモードが高くなり，3000付近では18.6と35.4で2倍近い差が出てくる。

実際のプログラムのコーディングでは，内積部分をサブルーチン化するか，インデックスを明示した形で直接インラインに展開するかのどちらかである。図3のように，スカラーモードではサブルーチン内積型，ベクトルモードではインデックス明示型が良いことがわかる。以上からS－810ベクトルモードで内積を使用する場合は，インデックスを明示した形で使用するのが良いことがわかる。

表3　スカイライン法（ISASⅡ/HAP）のCPU性能（S-810）
Table 3　CPU time in skyline method of ISASⅡ/HAP (S-810)

項番	自由度数	全体	剛性行列	三角分解	前進・後退	その他	実測値（秒）	HAPモード
1	2127	0.81	0.20	0.11	0.01	0.49	37.3	ノーペアモード
		0.81	0.20	0.11	0.01	0.49	37.3	ペアモード
		1.0	0.25	0.24	0.01	0.50	46.2	スカラーモード
2	4472	0.76	0.19	0.22	0.01	0.34	62.7	ノーペアモード
		0.76	0.19	0.22	0.01	0.34	62.5	ペアモード
		1.0	0.23	0.41	0.02	0.34	82.1	スカラーモード
3	7752	0.64	0.15	0.24	0.01	0.22	123.7	ノーペアモード
		0.63	0.15	0.24	0.01	0.23	122.9	ペアモード
		1.0	0.18	0.56	0.02	0.24	194.0	スカラーモード
4	18096	0.29	0.08	0.10	0.007	0.103	408.9	ノーペアモード
		0.28	0.09	0.09	0.007	0.093	393.3	ペアモード
		1.0	0.09	0.81	0.013	0.087	1414.3	スカラーモード

（注）各データはそれぞれスカラーモードでの全体CPU時間を1.0として
それに対する比で示した。実測値は全体CPU時間である。

5、S－810とスカイライン法

ISASⅡ/HAPで採用するスカイライン法を決定するため，ISASⅡのスカイライン法にS－810向けのコーディング修正をほどこして，S－810上で実測した結果を表3に示す。2000から8000自由度あたりまでは、ノーペアモード，ペアモードとも行列の三角分解のHAP加速率は1.9～2.3で，18000自由度では8.1～9.0となる。又，全体のCPU時間については1.6～2.3の加速率で，18000自由度では3.4～3.6の加速率となる。

このHAP加速率の値の妥当性を調べるため、George & Liu[1] のインコアスカイライン法と比較テストを行った。データとしては表3の項番1のデータを使用した。リージョンサイズとして6MBを割り当てた。ISASⅡ/HAPのスカイライン法がスカラーモード，ノーペアモード，ペアモードでそれぞれ三角分解に11.2秒，5.1秒，5.0秒を要した。これに対してインコアスカイライン法では，同じモード順で10.9秒，1.47秒，1.43秒となった。但し、インコアスカイライン法では，行列データのメモリへの読み込みは無視している。このデータの平均DOループ長は約60であり，表2からHAP加速率は7～8である。インコアスカイライン法の実測のHAP加速率が7.4～7.6は妥当である。それに対してISASⅡ/HAPの加速率は2.2と低い。これは行列の三角分解を純然たる内積演算とその他の前処理に分けたとすると，2000自由度では三角分解のうち、前処理に約45％ものCPU時間を費していることになる。これが18000自由度になると前処理時間の比は6％ぐらいに下がり、大部分が純然たる内積演算時間と考えられる。

ここで大切なことは、ベクトル化率を上げることはもちろんであるが，DOループ長をいかに長くするかが大切だということである。自由度数が増大するとHAP加速率が向上するのは，DOループ長が長くなることも寄与している。

前処理には二つある。一つはディスクファイルに圧縮して格納されている行列の非ゼロ成分をメモリに展開（アンパッキング）したり、メモリ上のデータを圧縮してディスクファイルに書き出す（パッキング）処理である。もう一つは行列のブロック特性を検索したり，アドレス計算したりする処理である。パックアンパックの平均CPU時間は一成分当り 2.3×10^{-7} 秒であり，２０００自由度では 0.92 秒，約１秒前後かかっている。ここで注意することは，パックアンパック時間が自由度数の２乗に比例してＣＰＵ時間がかかることで，１８０００自由度になると約７４秒で全体ＣＰＵ時間１２５秒の６０％を占めることである。

２０００自由度のデータでは純内積演算に１.５秒，パックアンパックに１秒かかるので残り２.５秒がブロック特性検索やインデックス計算に占める時間である。この時間は高々自由度数に比例するだけなので，１８０００自由度でも高々２３秒程度にしかならない。

内積演算のＣＰＵ時間 T_N は

$$T_N = \alpha_N \cdot \frac{m_N^2}{2} \cdot N \qquad (N：自由度数) \qquad (10)$$

で計算できる。ここで m_N は平均帯幅，α_N は単項当りのＣＰＵ時間（図３）である。$N=18000$ だと，$m_N=350$，$\alpha_N=3\times10^{-8}$ 秒として $T_{18000}=33$ 秒となる。

１８０００自由度では内積時間が３３秒，パックアンパックに７４秒，ブロック特性検索やインデックス，アドレス計算に２３秒，合計１３０秒かかると予測できる。これは実測値の１２５秒に近い。以上は２０００自由度の実測値から１８０００自由度の場合を推定した訳であるが，実測値に近いことから，この三者の比率はほぼここで推定した値と考えてよいだろう。

このように，汎用プログラムの三角分解法は純然たる演算だけでなく，データのパッキング，アンパッキングが最も大きな要因となることが理解できよう。それに加えて、内積以外のスカイラインの準備動作（ブロック特性検索やインデックス，アドレス計算など）もベクトル化されない部分であり無視できない部分である。

6. Ｓ－８１０を生かすには

Ｓ－８１０の特長を生かすためには

(i) ベクトル化率を高める。
(ii) 速い演算タイプを選ぶ。（内積演算，ストア演算等）
(iii) ＤＯループ長を長くする。
(iv) 並列演算器を有効に利用する。（ペアモード，ペアドスカイライン法）
(v) 拡張記憶を使用する。
(vi) リストベクトル処理を利用する。

上記項目が上げられる。

(i)のベクトル化率を上げることは，Ｓ－８１０に限らずすべてのスーパコンピュータについて言えることで，ベクトル化が適用できない部分が多いのでは，性能は上がらない。

(ii)はコーディングレベルだけの話ではない。ウェーブフロント法のように、フロント行列を作りながら中間結果をあとまで保持するやり方は自然とストア型の

演算になるし，スカイライン法のように現在計算しようとする列以前のデータしか参照しない方式では内積型の演算となる。オラフ・ルーベック[2]らのX－MP，VP－200，S－810のスーパコンピュータベンチマーク比較報告によると，S－810のストア型演算（$A(I)=B(I)\cdot C(I)+D(I)\cdot E(I)$）は内積型（$S=S+A(I)\cdot B(I)$）に比べて，DOループ長が10，50，100，200，1000の時，それぞれ，2.6倍，2.4倍，1.9倍，1.5倍，1.2倍の性能を持つという。X－MP，VP－200でも，ほぼ2倍前後の性能比を示しているが，S－810と比べて異なる点はDOループ長が1000になると，内積の方が若干有利になる点である。S－810ではDOループ長50のストア型演算が、ループ長150の内積型演算と同程度の性能を持つと考えてよい。ただし、同一の問題をスカイライン法とウェーブフロント法で解いた場合の平均ループ長は一般的にいってスカイライン法の方が長いようである。これは，スカイライン法がスカイラインの内部のゼロ成分も含めて演算するのに比べて，ウェーブフロント法がゼロ成分を除外して演算するためである。このようにスカイライン法にしてもウェーブフロント法にしても一長一短がある。

(iv)，(v)については7章，8章で述べる。

(vi)のリストベクトル利用については、インデックス計算を事前に配列に記憶してベクトル化をはかることができるかどうか検討の余地がある。

7. ペアドスカイライン法

並列演算器を有効に利用する一つの方法はS－810のペアモードを使用することである。ペアモードはDOループの中味を2個以上のステートメントに分けるというコンパイラのオプション機能である。さらにこの考えを押し進めて，2行同時にまとめて処理するペアドスカイライン法を採用することで並列処理の度合が高められる。ペアドスカイライン法のアルゴリズムは次のようになる。

$$u^*_{ij} = a_{ij} - \sum_{k=s}^{i-1} u_{ki} u^*_{kj} \quad (i=2,\cdots,j-1) \qquad (11)$$
$$u_{ij} = u^*_{ij} / d_{ii} \quad (i=1,\cdots,j-1) \qquad (12)$$
$$d_{jj} = a_{jj} - \sum_{k=s}^{j-1} u^*_{kj} u_{kj} \qquad (13)$$
$$u_{i+1j} = a_{i+1j} - \sum_{k=s}^{i-1} u_{ki+1} u^*_{kj} - u_{ii+1} u^*_{ij} \qquad (14)$$
$$u_{ij+1} = a_{ij+1} - \sum_{k=s}^{i-1} u_{ki} u^*_{kj+1} \qquad (15)$$
$$u_{i+1j+1} = a_{i+1j+1} - \sum_{k=s}^{i-1} u_{ki+1} u^*_{kj+1} - u_{ij+1} u^*_{ij+1} \qquad (16)$$

ここで$S=max(S_i, S_j)$で，それぞれS_iはi列と$i+1$列，S_jはj列と$j+1$列の最初の格納行番号である。ペアドスカイライン法はスカイライン法に比べてDOループの回数が半分になる。(11)式と(15)式を同時に計算するやり方を2列同時ペアドスカイライン法といい，(11)式，(14)式～(16)式を同時に計算するやり方を2行2列同時ペアドスカイライン法とよぶ。ペアドスカイライン法は2列単位で処理する。ペアとなる2列の非ゼロ成分の行番号は通常一致しないため，ベクトルの短かい方の列にゼロ成分を足して2列ずつの組を作成する。この方法をブロック化し、ブロックペアドスカイラインプログラムを試作してノーペアモードでテストした結果を表4に示す。2行2列同時ペアドスカイライン法はスカイライン法に比べて、ベクトルモードで約2倍の効果が期待できることがわかった。但し、ここではプロフィールがスカイライン法に比べてゼロ成分の追加分だけ数%程度

表4　ペアドスカイライン法の効果（ノーペアモード）

Table 4　Effects of paired Skyline method (no-pair mode) (単位:秒)

自由度数	2行2列同時ペアドスカイライン法		2列同時ペアドスカイライン法		スカイライン法		HAPモード
	時間	対スカラライン加速率	時間	対スカイライン加速率	時間	対スカイライン加速率	
2127	0.51	10.4	0.65	8.27	1.04	5.11	ノーペアモード
	5.14	1.03	5.11	1.04	5.31	1.00	スカラーモード

（注）対スカイライン加速率　はスカイライン法（スカラーモード）に対する性能比で示した。

の増加が見られる。ペアモードにした場合は最内側のDOループが2分され，しかもそれぞれに対しベクトルレジスタコンフリクトが生じ，スカイライン法より50％近く遅くなった。したがってISASII／HAPではノーペアモードのペアドスカイライン法を採用する予定である。

8．半導体拡張記憶装置の利用

S－810ではディスクの代りに大容量半導体記憶装置を使用することができ，従来のディスクアクセスの150～300倍のアクセススピードを持つ。拡張記憶は3GBまでの一時的ファイルを扱うことができるので，スカイライン法に現われる行列はほとんどこの中に納まってしまう。これによりI／Oネックは解消され，CPU時間とほぼ等しい時間，例えば18000自由度ならば5～6分で解けることになる。ISASII／HAPでは直接探索方式のFORTRANのI/Oを採用していくことで拡張記憶を有効に利用することを考えている。

9．むすび

われわれはスーパコンピュータS－810を有効に汎用構造解析プログラムISASII／HAPに適用する方式を検討して来た。ペアドスカイライン法，拡張記憶といった従来にない方式でスーパコンピュータにふさわしいソルバーを持たせる見通しがついた。インデックス処理のリストベクトル化，拡張記憶方式をサポートする際のアクセス方法の変更等，今後解決しなければならない問題もあるが，十分な検討を重ねた上でISASII／HAPに取り込んでいく予定である。

参考文献

1) Alan George & Joseph W. Liu : Computer Solution of Large Sparse Positive Definite Systems, Prentice-Hall, Inc., Englewood Cliffs, N.J (1981)

2) オラフ・ルーベック，ジェイムズ・ムーア，ラウル・メンデス：富士通，日立，クレイのスーパーコンピュータをベンチマークで比較，日経コンピュータ，1985.9.2，151～166 (1985)

3) 村田健郎，小国力，唐木幸比古：スーパーコンピュータ／科学技術計算への適用，丸善 (1985)

4) S. Harano : Improvements in Sparse Matrix Operations of NASTRAN, NASA CP-2151, 14～48 (1980)