ハイパフォーマンス
コンピューティング 57−14
（1995. 8. 24）

階層化サブストラクチャ法の適用による有限要素法の並列化とその実装

福盛秀雄　河野洋一　西松 研　村岡洋一

早稲田大学 理工学部
〒169 東京都新宿区大久保 3-4-1

fukumori@muraoka.info.waseda.ac.jp

本発表では、階層化サブストラクチャ法を利用した有限要素法の並列化と富士通の並列計算機 AP1000 への実装について述べる。階層化サブストラクチャ法は領域分解と直接法を組み合わせた並列化手法である。大規模疎行列の代わりに多数の小規模密行列を独立に計算処理するため、他の並列化手法と比較すると並列化が容易であり、かつ実行時間の点でも優れているという特徴を持っている。さらに本実装ではサブストラクチャ階層化テーブルの採用により、全体の並列化効率の向上を実現しようとしている。

Parallelization of FEM Using Multi-level Substructure Method

Hideo Fukumori　Yoichi Kono　Ken Nishimatsu　Yoichi Muraoka

School of Science and Engineering, Waseda University
3-4-1 Ohkubo Shinjuku-ku, Tokyo 169, JAPAN.

We implemented multi-level substructure method for FEM and its on the Fujitsu AP1000 parallel computer. This method first decomposes the domain into small substructures. These substructures make transformed coefficient matrices for the nodes on the substructure boundary and these matrices are added up in pairs to form larger substructures. This process is repeated until there is only one substructure left.
Size of matrices processed in each substructure is small and this helps to reduce overall computational time. In addition, each matrix can carry out most of the matrix transformation independent of other substructures, which helps to achieve high parallelism.

1 はじめに

本発表では、階層化されたサブストラクチャ法の並列計算機への実装について論じる。

有限要素法は数値計算の世界において重要な位置を占める手法であり、その並列計算機への実装は従来より数多く試みられている。

有限要素法の並列化のほとんどは

1. 有限要素法で作成された係数行列に対する並列化 [1],[2],[3]

2. 解析領域を分解し、各部分領域にプロセッサを割り当て、反復法による求解を実行する並列化 [4],[5],[6]

のいずれかに分類される。しかし 1. においては、処理の粒度の小ささによるプロセッサ間通信の多さが、並列化効率の向上において妨げとなる。また、2. においては、一回の反復毎に各部分領域で同期をとらなければならないため、部分領域によって収束の度合が異なる場合、やはり並列化効率は低下してしまう。さらに 2. による並列化では、対象とする問題によって異なる定式化を行なわなければならない。

実際には上に示した、2 つの主な方法の他にも、有限要素法の並列化手法は存在する。サブストラクチャ法はその一つである [7][8]。この方法は解析領域の分解と直接法による求解を組み合わせたものである。サブストラクチャ法は解析しようとする領域をサブストラクチャと呼ばれる小領域に分解し、これら小領域について各々連立方程式を生成する。これらを代入操作によって、サブストラクチャの外周部に存在する節点のみについての式に変形した後、足し合わせ処理によって、スーパストラクチャと呼ばれる、各サブストラクチャの外周節点の集合で構成される部分についての解を求める。その後に各サブストラクチャ内部の節点についての解を求めるものである。

サブストラクチャ法において内部節点を求める計算はサブストラクチャ間で完全に独立に実行可能なため、この部分においては、非常に高い並列性が実現できる。

しかしサブストラクチャ法については、以下のような問題点がある。

- スーパストラクチャの求解においては、改めて異なる並列化手法を実装しなければならず、この部分において全体の並列性は低下することになる。
- 計算効率の観点から見ると、サブストラクチャ数を増やしてゆくことで内部節点の計算過程における fill-in を減らしてゆくことが可能となるが、この場合スーパストラクチャの節点についての係数行列がその分大きなものとなり、そこで新たな fill-in が発生してしまう。

今回提案する階層化サブストラクチャ法は並列性と計算効率およびメモリ利用効率の向上を同時に実現しようとするものである。この方法では足し合わせの際にスーパストラクチャのための大きな係数行列を一つ生成する代わりに、二つ一組で足し合わせを実行しスーパストラクチャを生成する。それらを一階層上位のサブストラクチャと見なして再び行列の変形と足し合わせを二つ一組で実行する。このような階層的計算処理を繰り返し実行してゆくことで、最初のサブストラクチャをできるだけ小さく取り、全体の計算過程で生じる fill-in を減らすと同時に、スーパストラクチャの計算における並列性を向上させることが可能となる。

本研究では、この階層化されたサブストラクチャ法を富士通の並列計算機 AP1000 へ実装し、評価した。

2 階層化サブストラクチャ法

サブストラクチャ法という手法自体は、有限要素法の逐次求解のための方法として、古くより知られている。この方法は元来、計算機のメモリ上に収まらない程に大きなサイズの問題を領域分解により部分的に解いてゆくための技術として開発されたものである。以下にサブストラクチャ法の実行手順を示す。

1. 解析対象領域を、サブストラクチャと呼ばれる小領域に分解する。サブストラクチャの数は任意に選ぶことができるが、並列化をする場合にはプロセッサ台数分と等しくすることが多い。

2. 各サブストラクチャにおける節点は、サブストラクチャ領域内部に存在する節点と、サブ

ストラクチャ外周部に存在する節点の2種類に分けることができる。(図1) ここで各サブストラクチャにつき、内部節点を先に、外周節点を後にした新たな番号付けをし、x_iをサブストラクチャ領域内部にある節点についての解、x_bをサブストラクチャ外周部にある節点についての解とおくと、次のような連立方程式が成立する。

$$\begin{bmatrix} A_{ii} & A_{ib} \\ A_{bi} & A_{bb} \end{bmatrix} \begin{bmatrix} x_i \\ x_b \end{bmatrix} = \begin{bmatrix} b_i \\ b_b \end{bmatrix} \qquad (1)$$

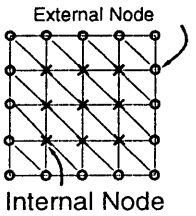

図 1: 内部節点と外周節点

3. 連立方程式 (1) の上半分の式を変形して得られるx_iを、下半分の式に代入すると、次のような、x_bのみについての式が得られる。

$$A^*_{bb}\, x_b = b^*_b \qquad (2)$$

ここで、

$$A^*_{bb} = [A_{bb} - A_{bi}A_{ii}^{-1}A_{ib}], \qquad (3)$$

$$b^*_b = [b_b - A_{bi}A_{ii}^{-1}b_i] \qquad (4)$$

である。

4. 各サブストラクチャのA^*_{bb} and b^*_bは足し合わされ、サブストラクチャ外周部に存在する節点 (図2) についての連立方程式

$$A^{*(global)}_{bb}\, x^{(global)}_b = b^{*(global)}_b \qquad (5)$$

を構成する。この外周部節点のみで構成される部分をスーパストラクチャと呼ぶ。(図2)

5. 連立方程式 (5) を解くことにより、スーパストラクチャについての解x_bを求める。

6. サブストラクチャ内部の節点についての解x_iを次の式によって求める。

$$x_i = A_{ii}^{-1}[b_i - A_{ib}x_b] \qquad (6)$$

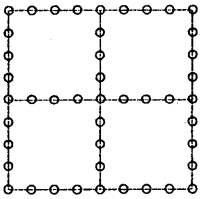

図 2: スーパストラクチャ

サブストラクチャの数を増やすと、各サブストラクチャについて作成される係数行列のサイズが小さくなるため、全体に占める fill-in が減少する。これにより計算過程における LU 分解に要する計算量を小さくすることができるが、反面スーパストラクチャについての係数行列のサイズは増大するため、この部分についての計算量は増大する。したがってサブストラクチャ数をむやみに増やしても全体の計算効率は低下してしまう。

この問題に対する解決法としては、サブストラクチャからスーパストラクチャへの足し合わせを階層的に実行してゆく方法が考えられる。最初にできるだけ大きな数のサブストラクチャを用意し、それらについての計算を実行した後、二つ一組でサブストラクチャの係数行列の足し合わせ処理を実行する。結果として最初に用意されたサブストラクチャの半数分のスーパストラクチャが新たに生成されることとなる。次はこれらの新たに生成されたスーパストラクチャをサブストラクチャと見なし、これらについて、あらためて係数行列の変形と足し合わせを二つ一組で実行し、その半数のスーパストラクチャを生成する。これを繰り返してゆくのが階層化サブストラクチャ法の考え方である。

階層化サブストラクチャ法の実行の手順は以下のようになる。

1. 解析しようとする領域を、内部に1節点を持つ、9節点から構成されるサブストラクチャに分解する。これを第1階層のサブストラクチャとする (図3)。

2. 第k階層のサブストラクチャにおいて：

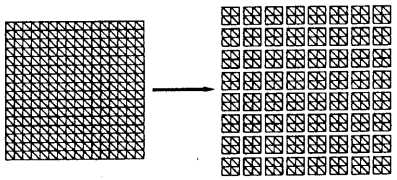

図 3: 第一階層サブストラクチャへの分解

(a) 通常のサブストラクチャ法と同様に計算を実行し、

$$\begin{bmatrix} A_{ii}^{(k)} & A_{ib}^{(k)} \\ A_{bi}^{(k)} & A_{bb}^{(k)} \end{bmatrix} \begin{bmatrix} x_i^{(k)} \\ x_b^{(k)} \end{bmatrix} = \begin{bmatrix} b_i^{(k)} \\ b_b^{(k)} \end{bmatrix} \tag{7}$$

より、

$$A_{bb}^{*(k)} x_b^{(k)} = b_b^{*(k)} \tag{8}$$

を導く。

(b) 各サブストラクチャは、隣接するサブストラクチャとペアを作り、一段階上の階層を構成するスーパストラクチャをを新たに構成する。これらを新たな階層におけるサブストラクチャと見なすこととする。

(c) $A_{bb}^{*(k)}$と$b_b^{*(k)}$は、ペアの相手であるサブストラクチャのそれと足し合わされることで、新たに連立方程式$A^{(k+1)}x^{(k+1)} = b^{(k+1)}$を構成する (図 4)。

3. サブストラクチャ数が 1 になるまで、上の手順を繰りかえす。

4. サブストラクチャ数が 1 になったら、すなわち最上位サブストラクチャに到達したら、$x_i^{(k)}$を求める。

5. 第 $k-1$ 階層のサブストラクチャは、$x_i^{(k)}$を利用して $x_i^{(k-1)}$を求める。

6. 上の手順を、k=1 になるまで、すなわち最下位のサブストラクチャに到達するまで繰り返す。

各階層で生成される行列はすべて密行列であるため、LU 分解などの過程における fill-in が大きく削減される。したがって計算効率およびメモリの使用効率が向上するという利点がある。

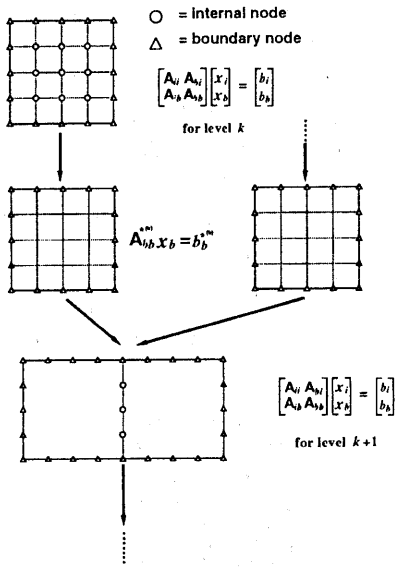

図 4: サブストラクチャの階層化

並列化の観点からは、各サブストラクチャにおける LU 分解等の行列計算は完全に独立に実行できるため、完全に並列実行が可能である。サブストラクチャ間でデータのやりとりが必要となるのは、二つ一組で上位階層を作る際の足し合わせ処理においてのみである。

3 階層化サブストラクチャ法の並列計算機への実装

今回は富士通の分散メモリ型並列計算機AP1000に階層化サブストラクチャ法を実装した。

解析形状とメッシュ分割にかかわらない柔軟な解析を実現するために、階層化テーブルを計算処理の実行前にホストプロセッサにて作成している。階層化テーブルは各サブストラクチャの親に当たるスーパストラクチャの番号、および節点番号の付け替え情報より構成されている。

下位のストラクチャ階層においては、サブストラクチャ数がプロセッサ数よりも多く、それぞれのサブストラクチャにおける計算は独立に実行可能であるため、完全な並列化が実現できる。

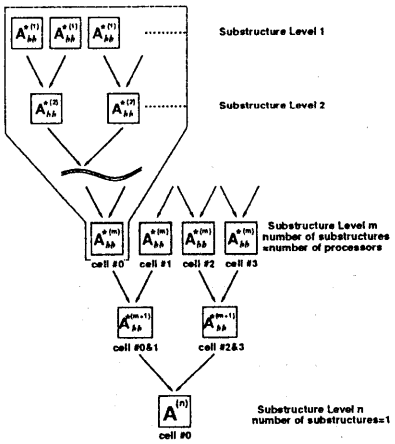

図 5: AP-1000 上における実装 (セル台数が 4 台の場合)

上位のサブストラクチャ階層においては、サブストラクチャ数がプロセッサ数よりも少なくなる。したがって 1 サブストラクチャに 1 プロセッサを割り当てるような単純な実装では上位階層での稼働プロセッサ数が減少してゆき、並列化効率が悪化してしまう。

それぞれのサブストラクチャにおける、A^*_{bb} を求める処理 (式 3) は、実際には以下のような手順で実行される。

$$A_{ii} = LL^t, \tag{9}$$

$$LM_{ib} = A_{ib} \rightarrow M_{ib}, \quad Lu_i = b_i, \quad L^t z_i = u_i, \tag{10}$$

(M_{ib} は $A_{bi}A_{ii}^{-1}A_{ib}$ を計算するための中間行列)

$$A^*_{bb} = A_{bb} - M^t_{ib}M_{ib} \tag{11}$$

これらのうち、式 (11) は M_{ib} を行単位でサイクリックに分割することによって、並列化が実現できる。

今回の実装では上位階層のサブストラクチャにおいては複数のプロセッサから構成されるクラスタを、それぞれのサブストラクチャに割り当て、式 (11) を並列実行することによって全体の並列化効率を向上させている。

4 性能評価

64×64 (節点数 4225, 三角形要素数 8192)、に分割された正方形領域に対するラプラス問題について、性能評価を行なった。

各問題におけるサブストラクチャ階層数は、64×64 メッシュの場合 12 階層であった。

表 1: 64×64 メッシュ分割における実行時間, 高速化率, および並列化効率

台数	時間 (秒)	高速化率	並列化効率 (%)
1	5.727	1.00	100.0
2	2.935	1.95	97.5
4	1.600	3.61	90.2
8	1.053	5.43	67.9
16	0.779	7.35	45.9
32	0.643	8.91	27.8
64	0.616	9.30	14.5

階層化サブストラクチャ法と通常のサブストラクチャ法 [8]、および共役勾配 (CG) 法 [3] の AP1000 上における実行時間を比較したものを表 2に示す。共役勾配法の実装のみ 1024 節点の場合での計測となっている。この表より、fill-in の削減による計算時間の短縮化の効果を確認することができる。

図 6に、64×64 メッシュ分割の場合における、プロセッサ台数に対するサブストラクチャ階層ごとの実行時間の変化を示す。プロセッサ台数が大きくなるに従い、上位 4 階層が全体の実行時間に占める割合が大きくなっている。これが表 1 における、大きなプロセッサ台数構成における並列化効率の低下につながっている。

上位の階層では先に述べた通り A^*_{bb} を求める処理のうち、式 (11) の並列化を行なっているが、これだけでは十分な並列化効率の向上は実現できて

表 2: 実行時間における他の並列化手法との比較

台数	階層化 節点数: 4096	オリジナル 節点数: 4096	CG 法 節点数: 1024
1	5.73	37.37	6.35
4	1.60	55.59	2.38
16	0.78	2.98	1.58
64	0.62	1.34	2.34

いない。しかし今回取り上げた問題の場合、上位の階層で扱われる行列のサイズは最大でも 64×64 であり、式 (9) のような LU 分解等の並列化を実装するには小さ過ぎる。より大規模の問題において行列計算に対し新たな実装を行ない、評価することが必要と考えられる。

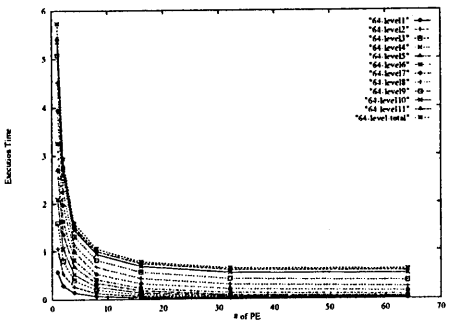

図 6: サブストラクチャ階層ごとによる実行時間の比較

5 まとめ

本発表では階層化サブストラクチャ法による有限要素法プログラムの並列化について述べた。サブストラクチャ法は、他の手法との比較において、実行時間において良い結果を示していることが確認された一方、上位のサブストラクチャ階層における並列化効率についてはさらに改善の余地があるということが明らかとなった。

今後の課題について以下に述べる。

- 上位サブストラクチャ階層での行列計算を現在の一部並列化から完全並列化に変更し、評価する。
- 三次元問題への対応。階層化テーブルの変更によって、計算部への変更なしにそのまま対応できるものと期待される。
- 不規則な形状、メッシュ分割を持つ問題において起こりうる負荷の不均等についての評価。

謝辞

並列計算機 AP1000 の利用にあたり、環境を提供してくださった (株) 富士通研究所に深く感謝いたします。

参考文献

[1] 加納 健, 中田 登志之, 他: "並列マシン Cenju 上の有限要素法による非線形変形解析", 並列処理シンポジウム JSPP'92, pp.399-406, 1992

[2] 加納 健, 中田 登志之, 他: "並列マシン Cenju2 上の有限要素法による非線形変形解析", 並列処理シンポジウム JSPP'93, pp.379-386, 1993

[3] 鶴見敬之: "有限要素法ソルバアルゴリズムの分散メモリ型並列計算機 AP-1000 への実装方法の研究", 村岡研究室 1993 年度修士論文, 1993

[4] G.Yagawa, N.Soneda and S.Yosimura: "A Large Scale Finite Element Analysis Using Domain Decomposition Method on A Parallel Computer", Computers & Structures Vol.38, No.5/6, pp.615-625, 1991

[5] 矢川元基, 曽根田直樹: "計算力学と CAE シリーズ 7 パラレルコンピューティング", 培風館, 1991

[6] G.Fox, M.Johnson, et al: "Solving Problems on Concurrent Processors (Volume1)", Prentice Hall, 1988

[7] I.St.Doltsinis and S.Nolting: "Studies on parallel processing for coupled field problems", Computer Methods in Applied Mechanics and Engineering vol.89, pp.497-521, 1991

[8] Hideo Fukumori and Yoichi Muraoka: "Parallel FEM Solution Based on Substructure Method", PCW'93 Proceedings of Fujitsu Second Parallel Computing Workshop, P1-K, 1993

[9] F.J.Peters: "Sparse Matrices and Substructures", Mathematisch Centrum, 1980