領域分割による並列 AMG アルゴリズム

藤井昭宏[†] 西田晃[†] 小柳義夫[†]

本稿では，Smoothed Aggregation に基づく Algebraic Multigrid 前処理付 CG 法（AMGCG 法）の並列アルゴリズムを提案する．特にアグリゲート生成後の次レベルデータ構造の生成を中心にアルゴリズムを考案する．データ構造に大きく依存するため，よく使われるであろうデータ構造を規定しその上でアルゴリズムを定義する．数値実験としてクラスタ上で最大 1562 万次元（$250 \times 250 \times 250$）の 3 次元ポアソン方程式を解き，ICCG 法（Localized ILU 前処理付 CG 法）との比較を行った．その結果，大規模な問題では計算時間で 3 倍以上有効であった．

A Parallel AMG Algorithm Based on Domain Decomposition

Akihiro Fujii,[†] Akira Nishida[†] and Yoshio Oyanagi[†]

In this paper, we have developed a parallel algorithm for Algebraic Multigrid method based on Smoothed Aggregation. We concentrate on data creation of the next coarser level after aggregate creation. We have tested three-dimensional Poisson problems with up to 15 million nodes ($250 \times 250 \times 250$) on a Myrinet cluster, and we have analyzed our method by comparing with ICCG (Localized ILU preconditioned CG mehod). The proposed method is about three times as efficient as ICCG in execution time on the large scale problem with 15 million nodes.

1. はじめに

近年，楕円型二階の偏微分方程式を離散化した大規模連立一次方程式 $Ax = b$ の反復解法としてマルチレベルな解法が多く研究されている．そのような手法の 1 つに Algebraic Multi-Grid 法（AMG）法がある．AMG 法は問題行列から，次元数の異なる複数の行列を生成して解く手法である．AMG 法の特徴として以下の 5 つがあげられる[1]．

- 逐次計算の場合，問題行列サイズを $n \times n$ とすると AMG 法がうまく機能すれば収束までの計算量は $O(n)$ である．
- 並列性がある．
- 不規則な疎行列に対しても適用可能である．
- 異方性問題に対しても有効である．
- 行列データしか参照しない．

AMG 法の中の最も有力な解法の 1 つに Smoothed Aggregation Multigrid[2)~4)] 法がある．Aggregation を並列に行う手法については多くの研究[5),6)] がなされているが，Aggregate 生成後，通信テーブルや次レベルの問題行列を並列に生成するアルゴリズムについて扱っている論文[7)] は少ない．そのアルゴリズムは適用するデータ構造に大きく依存する．

そこで本稿では大規模問題に対して代表的なデータの分散方法を規定し，その上での Smoothed Aggregation に基づく AMG 法を並列化するアルゴリズムを提案し評価を行う．

数値実験では問題サイズに対する収束時間，反復回数の変化を評価する．大規模なポアソン方程式（最大 1500 万次元）に対して GeoFEM ライブラリ[8)] に含まれる ICCG ソルバ（Localized ILU 前処理付 CG 法）[9)] と性能比較を行い，有効性を検証する．最後に本アルゴリズムの課題を検討する．

2. Smoothed Aggregation MG 法

Smoothed Aggregation MG 法は AMG 法の 1 つである．本稿が対象とする問題行列は楕円型偏微分方程式を離散化した対称正定値，既約行列とする．以下にこの解法の特徴，アルゴリズムをまとめる．

2.1 関連手法との関係

多くの定常反復解法は誤差の高周波数成分を取り除くことはできるが，低周波数成分を残してしまい収束が遅くなる．マルチグリッド（MG）法は誤差の低周波数成分も高周波数成分と同様に取り除くことができ

† 東京大学大学院情報理工学系研究科コンピュータ科学専攻
Department of Computer Science, Graduate School of Information Science and Technology, The University of Tokyo

MG 法 複数のレベルで固有の行列を生成.
- **Geometric MG 法** メッシュを生成して離散化を繰り返す.
- **Algebraic MG 法** 問題行列のみから次の粗いレベルの行列を生成．行列生成の方法により，Schur complement の近似を用いるマルチレベル ILU と，行列積 P^TAP を用いるその他の AMG 法に分けられる.

図 1 Geometric MG 法と Algebraic MG 法（AMG 法）
Fig. 1 Geometric and Algebraic MG method.

る定常反復解法である.

MG 法は問題方程式を複数のレベルに離散化して解く手法である．それぞれのレベルに対して次元数の異なる固有の問題行列が生成される．複数の問題行列を利用することにより，それぞれの離散化レベルに対応した誤差周波数成分を取り除くことができる．すなわち，反復ごとにすべてのレベルの問題行列を使うことで，1 反復で誤差の高周波数成分から低周波数成分まで均等に取り除くことができる．その結果，収束までの反復回数が問題サイズに非依存の解法となる.

MG 法では各レベルの行列の作成方法により Geometric MG 法と Algebraic MG（AMG）法に分類される．Geometric MG 法では，レベルごとにメッシュを作り直して離散化を繰り返し，行列を生成する．問題を解く際にメッシュの情報も処理する必要があり，有限要素法などによる不規則メッシュでは扱いが困難になる．一方，AMG 法は，問題行列のデータのみから各レベルごとに行列を生成する．必要なデータは問題行列のみであり，不規則疎行列にも適用できる.

AMG 法の中には，あるレベルの行列 A から粗いレベルの行列を生成する際に，Schur complement を利用する方法とレベル間の補間行列 P により P^TAP により算出する方法がある．Schur complement を用いる手法はマルチレベル ILU ともいわれる．Smoothed aggregation MG 法は P^TAP により行列を生成する AMG 法である.

図 1 に，Geometric MG 法と Algebraic MG 法（AMG 法）の特徴をまとめる.

2.2 Smoothed Aggregation MG 法のアルゴリズム

AMG 法は問題行列の階層構造の生成部（問題行列生成部）と，それを用いての反復解法部に分けられる．AMG 法の 1 種である Smoothed Aggregation のアルゴリズムを**図 2** に示す．ただし，$LEVEL$，

```
/* Input : A_1, x_1, b_1 */
/* 問題行列生成部 */
/* A_1 から A_2, ..., A_LEVEL と */
/* P_2, ..., P_LEVEL が作られる */
for lev = 1 to LEVEL − 1 do
  clev = lev + 1
  /* フィルタリング */
  Ã_lev = filter(A_lev)
  /* アグリゲート生成 */
  P̃_clev = aggregation(Ã_lev)
  /* アグリゲートの重み付 */
  P_clev = smooth(P̃_clev)
  A_clev = P^T_clev A_lev P_clev
end for

/* 反復解法部 (v cycle) */
/* S_lev はレベル lev での*/
/* ヤコビ法などの緩和法 */
for itercnt = 1 to MAXCOUNT do
  for lev = 1 to LEVEL − 1 do
    clev = lev + 1
    S_lev(A_lev, b_lev, u_lev)
    b_clev = P^T_clev(b_lev − A_lev u_lev)
  end for
  S_LEVEL(A_LEVEL, b_LEVEL, u_LEVEL)
  for lev = LEVEL − 1 to 1 do
    clev = lev + 1
    u_lev = u_lev + P_clev u_clev
    S_lev(A_lev, b_lev, u_lev)
  end for
end for
```

図 2 Smoothed Aggregation MG アルゴリズム
Fig. 2 Algorithm of Smoothed Aggregation MG.

$MAXCOUNT$ はそれぞれ最大レベル数と反復回数を表す定数．各レベルの緩和法 S_{lev}，問題行列 A_{lev}，右辺ベクトル b_{lev}，解 u_{lev}，レベル lev からレベル $lev-1$ への補間（prolongation）演算子 P_{lev} とする．lev が大きいほど，粗いレベル，すなわち対応する問題行列の次元数が小さくなるとする.

はじめに，問題 $A_1u_1 = b_1$ が与えられる．問題行列生成部では，粗いレベルを次々生成し，行列の階層構造を作成する．反復解法部ではそれを用いて V サイクルを行う.

問題行列生成部では A_{lev} から次のレベルの行列 A_{clev} を生成する．A_{lev} をフィルタリングし，そのグラ

フによりアグリゲートを生成する．その後，アグリゲート情報からレベル間（レベル $clev$ とレベル lev）の補間行列 P_{clev} を生成し，行列積 $A_{clev} = P_{clev}^T A_{lev} P_{clev}$ を計算する．

フィルタリング 問題行列 A の非ゼロ要素のうち値が小さいものは無視し

$$\tilde{A}_{lev} = filter(A_{lev})$$

を計算する．$filter()$ は，行列を引数として対角に対して絶対値の小さい非対角要素 $a_{ij}^2 < \theta^2 * |a_{ii}| * |a_{jj}|$ を 0 にする操作である．ただし θ は 0 から 1 の間の定数とする．数値実験では $\theta = 0.06$ としている．

アグリゲート生成 $\tilde{A_lev}$ に基づくグラフ上で考える．節点は $\tilde{A}_{lev}$ の各行，枝は非ゼロ要素に対応する．節点全体をアグリゲートと呼ばれる節点集合に分解する．アグリゲートは次の粗いレベルで 1 つの節点に対応し，グラフ上である節点を中心に近くの節点をまとめた節点集合と定義される．多くの場合ある節点から枝 1 本以内で到達できる節点集合となる．

以下の条件を満たすように全節点集合をアグリゲートに分けていく．

- 任意の節点がどこかのアグリゲートに属する．
- アグリゲートどうしは同じ節点を共有しない．

たとえば 5 点差分により生成された行列があったときはそれぞれのアグリゲートは 5 節点になる．ここで行列 $\tilde{P}_{clev}$ を作成．同じくらいの大きさのアグリゲートを整然とつくることができるかどうかによって収束性が変わる．

$$\tilde{P}_{clev}(i,j) = \begin{cases} 1 & \text{節点 i} \in \text{アグリゲート j} \\ 0 & \text{それ以外の場合} \end{cases}$$

アグリゲートの重み付け アグリゲートの節点に適切に重み付けをする．$\tilde{P}_{clev}$ をそのままレベル間の補間行列として使ってもよいが，収束性を高めるために緩和法を 1 回適用する，緩和法としては減速ヤコビ法を用いる場合が多い．減速ヤコビ法を用いた場合，ω を係数，D_{lev} を行列 A_{lev} の対角部として

$$P_{clev} = (I - \omega D_{lev}^{-1} A_{lev}) \tilde{P}_{clev} \qquad (1)$$

と書ける．

反復解法部 通常の MG 法と同様である．例としてレベル数が 2 のときを取り上げる．

はじめにレベル 1 で緩和法 $S_1(A_1, b_1, u_1)$ を行う．その残差をレベル 2 に送り，それに対応する補正解を計算する．すなわち $b_2 = P_2^T(b_1 - A_1 u_1)$ とし，レベル 2 での緩和法 $S_2(A_2, b_2, u_2)$ を計算する．u_2 にはレベル 1 の残差に対応する補正解が入っているので，この補正解をレベル 1 の解に足し込み反復解法を行う．つまり $u_1 = u_1 + P_2 u_2$ とし，レベル 1 で緩和法 $S_1(A_1, b_1, u_1)$ を行う．以上のような手順を繰り返すことにより，それぞれのレベルに対応する誤差周波数成分を同時に効率良く減らすことができる．

反復解法部で行うことは行列ベクトル演算とベクトルとベクトル加減算のみであり，並列性が高い．

3. データの分散方法

問題行列をグラフ構造として考える．節点は行列の各行，枝は行列の非ゼロ要素に対応する．

本研究では節点に基づく領域分割をしている．問題領域の有限要素集合を各 PE へ分割し，並列に有限要素集合を離散化して問題行列を生成する．生成された問題行列に対してソルバを適用する．ソルバ内部では同じ構造を持った問題行列が複数レベル分生成され，処理される．

以下に各 PE の担当する有限要素集合，問題行列を規定する．

担当節点集合 節点について PE それぞれに節点を重複なく分割する．問題領域内の全体節点集合 w_l をプロセッサ数 p 個の節点集合に分割したものを $w_{l,s}$, $s = 1, \ldots, p$ と定義する．すなわち

$$w_l = \bigcup_{s=1}^{p} w_{l,s},$$

ただし，$w_{l,s} \cap w_{l,q} = \phi, \forall q \neq s$, $s, q = 1, .., p$ となる．担当節点集合のサイズを $N_{l,s} = |w_{l,s}|$ と表す．l はレベルを，s は PE 番号を表す．

有限要素集合 担当節点集合 $w_{l,s}$ をノードとして含む有限要素を各 PE が持つ．全体の要素集合 f_l を p 個の有限要素集合 $f_{l,s}$, $s = 1, \ldots, p$ に分ける．

$$f_{l,s} = \{k | k \in f_l, \exists n \in w_{l,s} : n \in nodes(k)\}$$

nodes はある有限要素の節点集合を表すこととした．

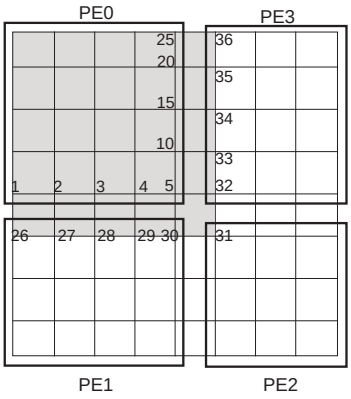
オーバラップレイヤと担当節点 プロセッサ s の有限要素集合 $f_{l,s}$ に含まれる節点集合を $W_{l,s}$ とすると $W_{l,s} \supset w_{l,s}$ となる．そのサイズを $NP_{l,s} = |W_{l,s}|$ と表す．すなわちプロセッサ s には $N_{l,s}$ 個の担当節点と $(NP_{l,s} - N_{l,s})$ 個のオーバラップしている節点がある．

問題行列 各 PE に割り当てられた有限要素集合を使い離散化する．各 PE の保持する行列サイズについては $NP_{l,s} \times NP_{l,s}$ となる．

PE 番号 s が持つ通信テーブルは，以下のようになる．

$neibPE(:)$ 隣接 PE 番号：$\{r \mid w_{l,s} \cap W_{l,r} \neq \phi\}$

$send(:, r)$ 隣接 PE それぞれに対する担当節点集合



$neibPE(:) = 1, 2, 3$
$send(:, 1) = 1, 2, 3, 4, 5$
$send(:, 2) = 5$
$send(:, 3) = 5, 10, 15, 20, 25$
$recv(:, 1) = 26, 27, 28, 29, 30$
$recv(:, 2) = 31$
$recv(:, 3) = 32, 33, 34, 35, 36$

図 3 PE0 が持つノード番号と通信テーブル
Fig. 3 Nodes and communication tables of PEO.

内の境界節点番号：$w_{l,s} \cap W_{l,r}$
$recv(:, r)$　オーバラップレイヤにある節点番号（外部節点番号）：$W_{l,s} \cap w_{l,r}$

たとえば**図 3** の場合，PE0 では

- 担当節点集合：節点番号 $1 \ldots 25$
- 外部節点集合：節点番号 $26 \ldots 36$
- 有限要素集合：灰色の部分
- 問題行列：36×36

となる．

4. 並列アルゴリズム

問題行列生成部の並列アルゴリズムを提案する．入力されるデータは問題行列と各 PE 間の通信テーブルのみである．はじめに $\tilde{A} = filter(A)$ の計算は並列化に影響しない．それ以降は以下の順で処理される．

- アグリゲート生成
- アグリゲートをスムージング
- 次レベル行列生成：行列積 $P^T AP$

以下，各ステップでの並列化について見ていく．

4.1 アグリゲート生成

並列にアグリゲートを生成する手法については，多くの研究[6),10)] がなされている．各 PE が担当節点集合を独立にアグリゲートを生成する手法や，最大独立点集合を並列に求めるアルゴリズム[11)] に基づく手法，

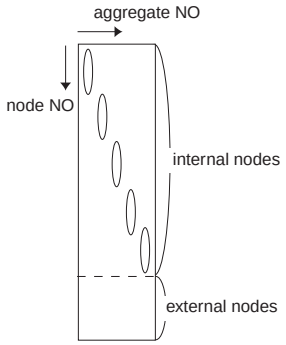


図 4 節点番号順にアグリゲートとなった場合の行列 $\tilde{P}$ の様子．図中の楕円は各アグリゲートを表す．楕円内の要素は 1 それ以外の要素は 0 となる
Fig. 4 Matrix $\tilde{P}$ with aggregates in node number order. A oval means non-zero elements.

領域間の境界部分をはじめに通信しあってアグリゲートを共有する手法などである．

並列アグリゲート生成アルゴリズムはアグリゲートの PE 間共有をするか，しないかで分類ができる．本稿では前者を共有アグリゲート生成，後者を独立アグリゲート生成と呼ぶことにする．本研究では独立アグリゲート生成の場合のみを対象にする．共有アグリゲート生成アルゴリズムの場合も本稿の並列化の拡張は可能である．

独立アグリゲート生成では，各 PE で担当節点集合内で独立にアグリゲートを持つことになる．たとえば各 PE が持つアグリゲートの行列 $\tilde{P}$ は**図 4** のようになる．行列 $\tilde{P}$ の各列が 1 つのアグリゲートに対応する．外部節点（external node）は，オーバラップレイヤにある節点集合である．

4.2 行列 P の生成

式 (1) に従って，行列 P を生成する．各 PE 独立に式 (1) を処理する．その後，行列積 $P^T AP$ を並列に計算するために，隣接 PE と行列 P の境界部分を交換する．以後この行列 P の境界部分の通信をアグリゲート情報の交換ということにする．どの PE からどのアグリゲートが渡されたかを外部アグリゲートテーブルに記録する．外部アグリゲートテーブルは PE 番号ごとにアグリゲート番号を配列として持つ．行列 P は**図 5** のようになる．外部アグリゲート（external aggregate）の列の部分が他 PE から通信で得た部分である．次にアグリゲート情報交換に関してまとめる．

アグリゲート情報の交換

各 PE k が $W_{l,k}$ の範囲で行列 P を正確に保持する必要があること，またアグリゲート情報の交換の際に通信テーブルを拡張することを以下に示す．

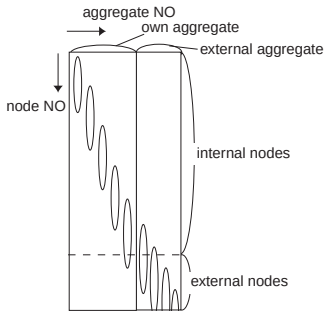


図 5 節点番号順にアグリゲートとなった場合の行列 P の様子．図中の楕円は各アグリゲートを表す．楕円内の要素は非ゼロそれ以外の要素は 0 となる
Fig. 5 Matrix P with aggregates in node number order. A oval means non-zero elements.

粗いレベルの行列 $A_c = P^T AP$ の計算を各 PE の担当節点集合に分けて行うこととする．行列 P の i, j 列目を v_i，v_j とすると，$A_c(i,j) = v_i^t A v_j$ となる．レベル l においてアグリゲート集合 G_l，アグリゲート $g_{l,i}$，関数 $\rho_l(i)$，集合関数 E を以下のように定める．

G_l 全 PE のアグリゲート集合
$g_{l,i} \in G_l$ i 番目のアグリゲート（節点集合）
$\rho_l(i)$ i 番目アグリゲートの担当 PE
E 節点集合に対して行列 A に基づくグラフ上で 1 層外側の節点も集合に入れる処理

アグリゲート $g_{l,i}$ に緩和法を適用し，アグリゲートはグラフ上で 1 層広がる．すなわち $E(g_{l,i})$ となる．$E(g_{l,i})$ はベクトル v_i の非ゼロ要素の節点に対応する．これを各 PE の担当節点集合に分割する．k 番目の PE が持つベクトルを $v_{i,k}$ とすると，その非ゼロ要素は

$$w_{l,k} \cap E(g_{l,i}) \tag{2}$$

の節点集合となる．また $v_i = \sum_{k=1,p}(v_{i,k})$ である．同様に $v_{i,k}^t A$ の非ゼロ節点集合は

$$E(w_{l,k} \cap E(g_{l,i})) \subset W_{l,k} \tag{3}$$

となる．行列積は

$$A_c(i,j) = \left(\sum_{k=1,p} v_{i,k}^t\right) A v_j = \sum_{k=1,p} (v_{i,k}^t A) v_j \tag{4}$$

のように並列化を行う．すなわち $A_c(i,j)$ を上記のように処理するには，各 PE ごとに v_i は式 (2)，v_j は式 (3) の範囲のデータが必要になる．式 (2) は式 (3) に包含されることから，k 番目の PE では行列 P は $W_{l,k}$ の範囲で正確に持つようにする．

PE 間で $W_{l,k}$ の領域が重なったところは，互いにアグリゲート情報を交換が必要である．各レベルで PEr が PEs に対して持つ通信テーブルは

$$neibPE = \{r \mid W_{l,s} \cap w_{l,r} \neq \phi\}$$
send: $w_{h,s} \cap W_{h,r}$ recv: $W_{h,s} \cap w_{h,r}$

である．ただし，$neibPE$ は隣接 PE 番号の集合，$send$，$recv$ は通信テーブルを表す．アグリゲート交換のときは以下のように別の通信テーブルを用意する．

$$neibPE = \{r \mid W_{l,s} \cap W_{l,r} \neq \phi\}$$
send: $W_{l,s} \cap W_{l,r}$ recv: $W_{l,s} \cap W_{l,r}$

この通信テーブルにより，アグリゲートの交換をすれば各 PE r が $W_{l,r}$ の範囲のアグリゲート情報を持たせることができる．

各 PE で $(v_{i,k}^t A)v_j$ の処理後，担当 PE$\rho_l(i)$ 上で和をとり $A_c(i,j)$ の処理が終了する．

4.3 行列積 $P^T AP$

この行列積は各担当領域内で独立に行う．すなわち，式 (4) の $\sum$ の内部を各 PE で処理する．PEk の担当領域 $w_{l,k}$ にある節点に対応する行だけ残してそれ以外の行を 0 にセットする処理を $M_k()$ とする．**図 6** 内の関数 $local\_rap$ は以下の処理をする．

$$local\_rap(A,p) : M_k(P^T)AP \tag{5}$$

次に，$sendrecv\_A()$（できた行列について外部アグリゲートに関する行を担当 PE に送り，担当 PE 上で足しこむ）を処理する．新しい外部アグリゲートがあった場合は外部アグリゲートテーブルに追加する．

ここで担当アグリゲートについての問題行列は完成する．$assign\_external\_nodes(A_{clev})$ で外部アグリゲートと担当アグリゲート間の値については対称行列であることを仮定して転置して代入する．$sendrecv\_table()$ では外部アグリゲートテーブルが粗いレベルの通信テーブル（RECV）となっているので，通信をして SEND 側も作成する．ここで粗いレベルの通信テーブルができる．その通信テーブルを使い $sendrecv\_D(A_{clev})$ により外部アグリゲートの対角要素は通信をして正しい値を格納する．

アルゴリズムの概略は図 6 のようになる．

5. 数値実験と評価

本研究で提案する並列 AMG アルゴリズムを CG 法の前処理として実装し評価する．比較のため GeoFEM[8] ライブラリのソルバと同じインタフェースで実装し，ICCG ソルバ[9] と比較を行った．問題サイズに対する反復回数や収束までの時間の変化に注目する．

問題は 3 次元ポアソン方程式で以下のものである．

$$-\left(\frac{\partial}{\partial x}\left(\frac{\partial P}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{\partial P}{\partial y}\right) + \frac{\partial}{\partial z}\left(\frac{\partial P}{\partial z}\right)\right) = 0$$

/* 問題行列生成部 */
for $lev = 1$ to $LEVEL - 1$ **do**
  /* $A_{lev}, send_{lev}, recv_{lev}$ から */
  /* $A_{clev}, send_{clev}, recv_{clev}, P_{clev}$ 作成 */
  $clev = lev + 1$
  $\tilde{A}_{lev} = filter(A_{lev})$
  $\tilde{P}_{clev} = aggregation(\tilde{A}_{lev})$
  $P_{clev} = smooth(\tilde{P}_{clev})$
  /* 通信テーブルの拡張 */
  $newtable = create\_table(send_{lev}, recv_{lev})$
  /* アグリゲート情報の交換 */
  /* 外部アグリゲートテーブル作成 */
  $sendrecv\_P(P_{clev}, ag\_table, newtable)$
  $A_{clev} = local\_rap(A_{lev}, P_{clev})$
  /* A_{clev} の外部アグリゲートに */
  /* 対応する行を通信．*/
  /* 外部アグリゲートテーブル更新 */
  $sendrecv\_A(A_{clev}, ag\_table)$
  /* A_{clev} の外部アグリゲートの行に*/
  /* 対称行列を仮定して代入．*/
  $assign\_external\_nodes(A_{clev})$
  /* 粗いレベルの通信テーブル作成 */
  $sendrecv\_table(ag\_table, send_{clev}, recv_{clev})$
  /* 外部アグリゲートの対角要素の更新 */
  $sendrecv\_D(A_{clev})$
end for

図 6 並列問題行列生成アルゴリズム：
$newtable$ はアグリゲート情報交換のために拡張した通信テーブル．ag_table は外部アグリゲートテーブル．$send_{lev}$ はレベル lev における SEND 通信テーブル．$\tilde{A}_{lev}$ は非ゼロ要素のうち対角と比較して小さい値を落とした行列

Fig. 6 Parallel algorithm of matrix creation:
$newtable$ is a communication table for exchange of aggregates, $send_{lev}$ is send table on level lev. $\tilde{A}_{lev}$ is filtered matrix by elements' value.

問題領域は X，Y，Z の各軸に垂直な面を持つ直方体とする．また境界条件については

$$\begin{cases} Xmin: & \frac{\partial P}{\partial x} = 10.0, \\ Ymin: & \frac{\partial P}{\partial y} = 5.0, \\ Zmax: & \frac{\partial P}{\partial z} = 1.0, \\ Zmin: & P = 0, \end{cases}$$

とし，その他の境界は自然境界条件とした．$Xmin$ は X 座標が最小の面を表す．1PE あたりの節点数を $50 \times 50 \times 50$ に固定して残差の 2 ノルムが初期残差の 10^{-13} 倍になるまでの時間を計測する．

AMG 法の反復解法部では各レベルで対称ガウスザイデル法 1 回を各 PE 領域内で行い，領域境界の節点についてはヤコビ法を行う．最も粗いレベルでは，この反復解法を 10 回行う．担当節点数が 70 点未満になる PE がでてくるまで次のレベルを生成するものとする．テスト環境には Sun Blade 1000（UltraSPARC III 750 MHz×2，主記憶 1 GB）128 ノードを Myrinet 2000 により接続したクラスタを用いた．ノード間通

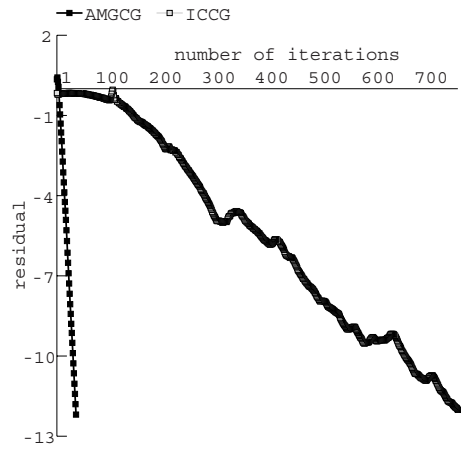


図 7 残差の変化：125PE, 250 × 250 × 250：縦軸は相対残差（2 ノルム）の対数，横軸は反復回数

Fig. 7 Relative residual: 125PE, 250 × 250 × 250: vertical axis is logarithm of 2-norm of relative residual, horizontal axis is number of iterations.

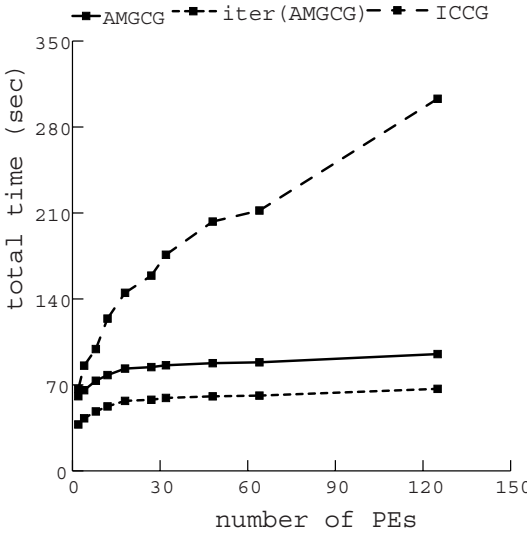


図 8 AMGCG と ICCG の実行時間：iter（AMGCG）は AMGCG の反復解法部にかかった時間

Fig. 8 Comparison of execution time between AMGCG and ICCG: iter (AMGCG) means time of iterative part of AMGCG.

表 1 AMGCG 法の計算時間と反復回数
Table 1 Execution time and iteration number of AMGCG.

PE 数	問題サイズ	反復回数	時間（秒）
2PE	50 × 50 × 100	25	60.8
4PE	50 × 100 × 100	28	65.7
8PE	100 × 100 × 100	31	73.4
12PE	100 × 100 × 150	33	78.0
18PE	100 × 150 × 150	35	83.3
27PE	150 × 150 × 150	35	84.5
32PE	100 × 200 × 200	36	86.0
48PE	150 × 200 × 200	36	87.7
64PE	200 × 200 × 200	36	88.4
125PE	250 × 250 × 250	37	95.1

表 2 AMGCG と ICCG の実行時間（秒）
Table 2 Execution time of AMGCG and ICCG.

PE 数	AMGCG	反復解法部（AMGCG）	ICCG
2PE	60.8	37.8	66.8
4PE	65.7	42.8	85.7
8PE	73.4	48.4	99.2
12PE	78.0	52.5	124
18PE	83.3	57.0	145
27PE	84.5	57.9	158
32PE	86.0	59.4	176
48PE	87.7	60.7	203
64PE	88.4	61.3	212
125PE	95.1	66.5	303

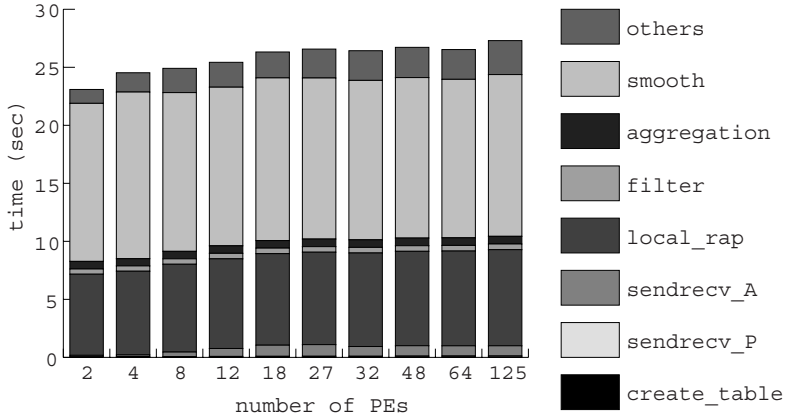


図 9 問題生成部内の構成：
図 6 内の対応する関数ごとに計測．積み上げグラフの下 3 つが主な通信時間である．ほとんどの時間は行列積（local_rap）とアグリゲートのスムージング（smooth）に費やされていることが分かる
Fig. 9 Each function's time in hierarchical matrix creation. Three elements in under part are main communication time. local_rap and smooth spend most of the execution time.

信は MPICH-GM [12]（version 1.2.1）により行った．以後，処理時間など，各 PE で値が異なるデータについては平均の値を書く．

図 7 は 125PE の場合の各手法の残差減少の様子を表した．**図 8** は PE 数を変化させた場合の実行時間を比較したものである．

図 7 では，AMGCG 法の残差は急速に減っている．この現象は MG 法による CG 法に対する前処理が非常に有効であることを示している．**表 1** では問題サイズによらず反復回数が一定であることが分かる．

AMGCG 法は並列化処理のオーバヘッドを除き AMG 法がうまく機能すれば，問題行列が $n \times n$ の場合，計算量 $O(n)$ の解法となる．図 8，**表 2** から今回のテスト問題については ICCG 法より有利なことが分かる．また大規模な問題においては AMGCG 法の優位性がより大きくなる．

$Ax = b$ の右辺だけが異なる問題を繰り返し解く必要がある場合は多いが，問題生成部は 1 度実行して前もって行列を作っておけばよい．その場合，反復解法部だけが実行されることになり，AMGCG 法はさらに有効な解法となる．

次に本研究で提案した並列化アルゴリズムの時間の内訳を見てみる．主な関数での時間の内訳を**図 9** に示す．

図 9 では，主な時間がアグリゲートのスムージング（関数 smooth）と行列積 RAP（関数 local_rap）に費やされていることが分かる．アグリゲートのスムージングは各 PE 独立にすることから，並列性に関係なくほぼ一定の時間で終了する．

次に行列積 RAP について考察する．問題行列生成部の 30 から 35% の時間が RAP の行列積（local_rap）に費やされている．また各 PE で問題サイズ固定で解いているにもかかわらず，PE 数が増えると行列積 RAP にかかる時間が増えている．PE 数が増えるに

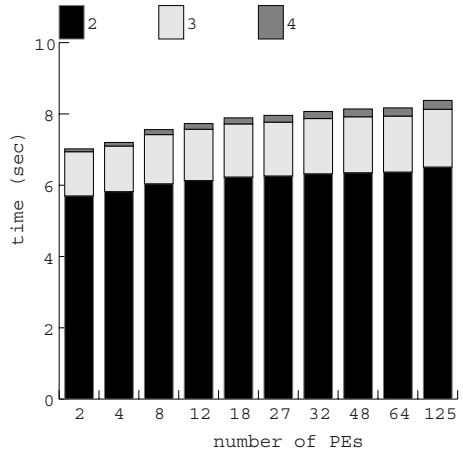


図 10 行列積 RAP のレベルごとの時間：
レベル 2 からレベル 4 まで行列積の処理時間．
通信時間は含まない
Fig. 10 Time of matrix product RAP for each level. It doesn't include communication time.

表 3 各レベルにおける担当節点の割合
Table 3 Rate of own nodes on each level.

PE 数	level1	level2	level3	level4
2PE	0.980	0.885	0.671	0.484
4PE	0.961	0.785	0.451	0.222
8PE	0.942	0.738	0.345	0.117
12PE	0.924	0.689	0.277	0.081
18PE	0.906	0.640	0.220	0.068
27PE	0.889	0.604	0.194	0.049
32PE	0.906	0.640	0.213	0.055
48PE	0.889	0.603	0.193	0.045
64PE	0.888	0.601	0.192	0.044
125PE	0.888	0.601	0.192	0.041

従って，特に粗いレベルで外部節点が増加しているためと考えられる．そのことは**図 10** により確認できる．

主な通信に関する時間（関数 create_table，sendrecv_P，sendrecv_A）は徐々に増えているが，ローカルに行列積 RAP の計算後に結果を担当 PE へ通信し足し込む時間（関数 sendrecv_A）がほとんどの時間を占めている．

これらの処理時間や通信時間は行列 A や P のデータ構造に大きく依存する．今後それぞれの処理に適切なデータ構造を考える必要があるかもしれない．

最後に今回の問題について各レベルで外部節点と担当節点の割合がどのように変化したかを**表 3** に示す．この数値実験では 4 レベル生成し，それぞれのレベルに対する自ノードの割合をまとめた．全 PE のうち，割合が最小のものの値を用いた．

この表から，並列性を高めれば高めるほど，粗いレベル（level3，level4）では担当節点の割合が減り，オーバラップ領域の節点が増え，通信時間や行列積の処理時間増加することが分かる．

通信時間（図 9）や AMGCG 全体の時間（図 8）は PE 数が $3 \times 3 \times 3$ 以上の並列度ではほぼ一定となっている．今回の実験では，1PE あたりの問題サイズが固定であり，他 PE との境界部分は最大 6 面であること，6 面とも他 PE に接する PE が出てくる問題は PE サイズ $3 \times 3 \times 3$ 以上であることによる．これは表 3 により，27PE 以上の並列度において担当節点の割合の減少が少なくなっていることからも確認できる．

各レベルでの担当節点数はどの PE 数によらずほぼ一定であった．2PE のときは 125000，7386，405，9 のように節点数が変化した．この場合，最後のレベルでは 9 節点しか残っていない．レベル数を 3 で切り上げた方が実行時間が短縮されるケースも多くあったが，今回は比較のためどの PE 数を適用する場合も同じレベル数を生成している．

6. おわりに

有限要素法により離散化される大規模問題に対して，節点に基づく領域分割のデータ構造を規定し，その上での Smoothed Aggregation MG 法の並列アルゴリズムを提案し評価した．

大規模な 3 次元ポアソン方程式において本手法は並列性が高く，また代表的な手法である ICCG 法と比較しても，より有効であった．反復解法部の処理時間は全体の処理時間のほぼ 2/3 であり，反復解法部のみを繰り返し適用するような問題に対してはさらに有効となることを確認した．

一方で問題行列生成部の時間の内訳を見ると，行列積 P^TAP とアグリゲートのスムージングにかかる時間の合計は問題生成部の約 85%も占めている．今後，アグリゲートのスムージングや P^TAP の高速化について考察する必要がある．また PE 数を増加して並列性を高めた場合に，粗いレベルでは担当節点数が減り外部節点数が増えている．これはそのレベルにおいて，処理時間の減少と通信時間の増大を示すものであり，より高並列な環境ではさらに広がるものと考えられる．そこで粗いレベルでの通信時間を少なくする工夫が必要となる．粗いレベルでは使用する PE 数を減少させることや，ハイブリッドプログラミングにより領域分割数を減少させることなどがあげられる．

今後上に述べたような改良や，領域により構造が異なるような行列に対して粗いレベルのロードバランスを考慮した問題行列生成アルゴリズム，ベクトル化などを考案することにより，大規模な対称正定値の疎行

列についてロバストな並列アルゴリズムを実現できると考えている．

謝辞 本研究を遂行するにあたり，高度情報科学技術計算機構の中島研吾氏に GeoFEM 向けテスト問題を提供していただきました．感謝いたします．なお，本研究の一部は，科学研究費補助金基盤研究（B）13480080 および科学技術振興事業団戦略的創造研究推進事業によるものである．

参 考 文 献

1) Cleary, A.J., Falgout, R.D., Henson, V.E., Jones, J.E., Manteuffel, T.A., McCormick, S.F., Miranda, G.N. and Ruge, J.W.: Robustness and Scalability of Algebraic Multigrid, *SIAM Journal on Scientific Computing*, Vol.21, No.5, pp.1886–1908 (2000).
2) Vanek, P., Brezina, M. and Mandel, J.: Convergence of Algebraic Multigrid Based on Smoothed Aggregation.
3) Vanek, P., Mandel, J. and Brezina, M.: Algebraic Multigrid by Smoothed Aggregation for Second and Fourth Order Elliptic Problems, Technical Report UCD-CCM-036 (1995).
4) Chan, T.F. and Vanek, P.: Multilevel algebraic Elliptic Solvers, *UCLA Math. Dept. CAM Report* (1999).
5) Cleary, A.J., Falgout, R.D., Henson, V.E. and Jones, J.E.: Coarse-Grid Selection for Parallel Algebraic Multigrid, *Workshop on Parallel Algorithms for Irregularly Structured Problems*, pp.104–115 (1998).
6) Henson, V.E. and Yang, U.M.: BoomerAMG: A parallel algebraic multigrid solver and preconditioner, *Applied Numerical Mathematics: Trans. IMACS*, Vol.41, No.1, pp.155–177 (2002).
7) Haase, G.: A parallel AMG for overlapping and non-overlapping domain decomposition, *Elect. Trans. Numer. Anal.*, Vol.10, pp.41–55 (2000).
8) 財団法人高度情報処理機構：GeoFEM. http://www.geofem.tokyo.rist.or.jp/
9) Nakajima, K. and Okuda, H.: Parallel Iterative Solvers with Localized ILU Preconditioning for Unstructured Grids on Workstation Clusters, *IJCFD* (1999).
10) Tuminaro, R.S. and Tong, C.: *Parallel Smoothed Aggregation Multigrid: Aggregation Strategies on Massively Parallel Machines*, pp.47–47 (2000).
11) Adams, M.: A Parallel Maximal Independent Set Algorithm, *Proc. 5th copper mountain conference on iterative methods.*
12) Myricom: Myrinet Software. http://www.myri.com/scs/

(平成 14 年 9 月 24 日受付)
(平成 14 年 12 月 11 日採録)

藤井 昭宏（学生会員）
1975 年生．1999 年東京大学理学部情報科学科卒業．2001 年東京大学大学院理学系研究科情報科学専攻修士課程修了．同年より東京大学大学院情報理工学系研究科コンピュータ科学専攻博士課程に在学．大規模線形問題に対するマルチレベルな解法に興味を持つ．

西田 晃（正会員）
1970 年生．1995 年東京大学理学部情報科学科卒業．1998 年東京大学大学院理学系研究科情報科学専攻博士課程修了．理学博士．同年より東京大学大学院理学系研究科情報科学専攻助手．2002 年より科学技術振興事業団戦略的創造研究推進事業「シミュレーション技術の革新と実用化基盤の構築」領域研究代表者を兼務．反復解法，特に大規模固有値解法と並列数値処理の研究に従事．ACM，IEEE，SIAM，日本応用数理学会，日本ソフトウェア科学会各会員．

小柳 義夫（正会員）
1943 年生．1966 年東京大学理学部物理学科卒業．1971 年東京大学大学院理学系研究科物理学専門課程修了，理学博士．同年東京大学助手．高エネルギー物理学研究所理論部門助手，筑波大学電子情報工学系講師，助教授，教授を経て，1991 年東京大学理学部情報科学科教授．並列処理，数値解析，計算物理学に関する研究に従事．特に，偏微分方程式の高速並列解法，最小二乗法の数値計算，乱数やモンテカルロ法に興味を持つ．物理学会，日本統計学会，応用統計学会，計算機統計学会，応用数理学会等各会員．