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 Method (AMG 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, GeoFEM や良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800 万次元の3 次元ポアソン方程式を解き, ICCG 法との比較を行ない本手法の有効性を示した. また本手法の処理時間の構成などを分析した.

Parallel AMG algorithm by domain decomposition

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

In this paper we developed a parallel algorithm of Algebraic Multigrid method based on Smoothed Aggregations. We concentrate on data creation of the next coarser level after aggregate creation. 3 dimensional Poisson problems which have 8 million nodes are solved with cluster computers in our numerical test, and we analyze our method by comparing that with ICCG and by the rate of communication time.

1. はじめに

近年, 楕円型2 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. そのような手法の一つにAMG 法がある. AMG 法の特徴として以下の4 つがあげられる.

- 逐次計算の場合, 収束までの計算量は$O(n)$ である
- 並列性がある
- 不規則疎行列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一つにSmoothed Aggregation Multigrid[2)3)] 法がある. Aggregation を並列に行う手法については多くの研究がなされているが, 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.

そこで本論文では大規模問題に対して代表的なデータ構造を規定し, 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なう.

数値実験では大規模なポアソン方程式 (最大800 万次元) に対してGeoFEM ライブラリ[7)] に含まれるソルバと性能比較を行ない, 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.

† 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
Dept. of Computer Science, Graduate School of Information Science and Technology, University of Tokyo

2. Smoothed Aggregation MG 法

マルチグリッド (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. 各レベルでは次元数の異なる問題行列が生成される. 複数レベルの問題行列を利用することにより, それぞれの次元数に対応した誤差周波数成分を取り除くことができる. その結果, 誤差周波数成分を均等に減らすことができ, 収束までの反復回数が問題サイズに非依存の解法になる.

Smoothed Aggregation Multigrid 法は問題行列の階層構造の生成部 (問題生成部) と, それを用いての反復計算部 (反復部) に分けられる. 図1 にアルゴリズムを書く.

但し $LEVEL, MAXCOUNT$ 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_{level}, 問題行列 A_{level}, 右辺ベクトル b_{level}, 解 u_{level}, レベル $level$ からレベル $level-1$ への補間演算子 (prolongation) P_{level} とする. また $level$ が大きいほど, 粗いレベルを表す.

2.1 問題行列生成部

$smoothed\_aggregation()$ では以下3つのことをする.

(1) 問題行列の非ゼロ要素のうち, 対角に比して値

```
/* 問題行列生成部 */
for level = 1 to LEVEL − 1 do
  clevel = level + 1
  P_clevel = smoothed_aggregation(A_level)
  A_clevel = P^T_clevel A_level P_clevel
end for

/* 反復解法部 (v cycle) */
for itercnt = 1 to MAXCOUNT do
  for level = 1 to LEVEL − 1 do
    clevel = level + 1
    S_level(A_level, b_level, u_level)
    b_clevel = P^T_clevel(b_level − A_level u_level)
  end for
  S_LEVEL(A_LEVEL, b_LEVEL, u_LEVEL)
  for level = LEVEL − 1 to 1 do
    clevel = level + 1
    u_level = u_level + P_clevel u_clevel
    S_level(A_level, b_level, u_level)
  end for
end for
```

図 1 AMG アルゴリズム

値以上のものを取り出す

(2) アグリゲート生成

(3) P_{level} 生成

(1) は, 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

$$\tilde{A} = filter(A)$$

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を 0 にする操作である.

(2) では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 つの節点となる. $\tilde{A}$ に基づくグラフ上で ある節点を中心に近くの節点をまとめた集合をアグリゲートと定義する. 多くの場合ある節点から枝 1 本以内でいける節点集合となる. 以下の条件を満たすように $\tilde{A}$ 上のグラフで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
- アグリゲート同士は重ならない

例えば 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 $\tilde{P}_{clevel}$ を作成.

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

(3) ではアグリゲートの節点に適切に重みづけをする. $\tilde{P}_{clevel}$ をそのままレベル間の補間行列として使ってもいいが, 収束性を高めるために緩和法を一回適用する. 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_{level} を行列 A_{level} の対角部として

$$P_{clevel} = (I - \omega D^{-1}_{level} A_{level})\tilde{P}_{clevel} \qquad (1)$$

と書ける.

2.2 反復解法部

通常の MG 法と同様. 2 レベルの場合：細かいレベルでの緩和法で残った残差を粗いレベルに移す. 粗いレベルでその残差を打ち消すような補正解を計算し, 細かいレベルへ補正解を足しこむ.

以上のような手順を繰り返すことにより, 誤差ベクトルのすべての周波数成分を同時に効率良く減らすことができる.

反復解法部で行な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. データ構造

本研究では節点に基づく領域分割をしている. 以下に各プロセッサが持っている節点データや問題行列を中心に定義する. 以下に示すデータ構造をすべてのレベルが持つことになる.

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_h をプロセッサ数 p 個の節点集合に分割したものを $w_{h,s}$, $s = 1, \ldots, p$ と定義する. すなわち

$$w_h = \bigcup_{s=1}^{P} w_{h,s},$$

但し $w_{h,s} \cap w_{h,q} = \phi$, $\forall q \neq s$, $s, q = 1, .., p$ となる. 担当節点集合のサイズを $N_{h,s} = |w_{h,s}|$ と表す. h 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 $w_{h,s}$ 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_h を p 個の有限要素集合 $f_{h,s}$, $s = 1, \ldots, p$ に分ける.

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

nodes はある要素の節点集合を表すこととした.

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

本研究では PEs が持つ通信テーブルは, 以下のようになる.

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

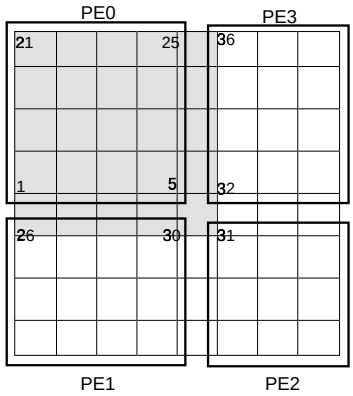
$send(:, r)$ 隣接 PE それぞれに対する担当節点集合内の境界節点番号: $w_{h,s} \cap W_{h,r}$

$recv(:, r)$ オーバーラップレイヤにある節点番号 (外部節点番号) : $W_{h,s} \cap w_{h,r}$

例えば図 2 の場合, PE0 では

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

となる.



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

図 2 PE0 が持つノード番号と通信テーブル

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. 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
- 行列積 $P^T AP$

アグリゲート生成後, アグリゲートのスムージングにより各PE は行列 P を算出する. 各PE が行列 P どの範囲までを持てば良いか考察する. 粗いレベルの行列を A_c とする. $A_c = P^T AP$ の計算を各PE の担当節点集合に分けて行なうことにする. 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.

全PE のアグリゲート集合 G_h, その i 番目のアグリゲート（節点集合）を $g_{h,i} \in G_h$, そのアグリゲートの担当PE を $\rho_h(i)$ とする. アグリゲート $g_{h,i}$ に緩和法を適用し, アグリゲートは1層広がる. すなわち $E(g_{h,i})$ となる. 但し E を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と定義する. $E(g_{h,i})$ 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

$$w_{h,k} \cap E(g_{h,i}) \quad (2)$$

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

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

となる. 行列積は

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

のように並列化を行なう. 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 $W_{h,k}$ の範囲で正確に持つようにする.

PE 同士 $W_{h,k}$ の領域のデータを交換できるように通信テーブルの作り替えをする. データ構造としてある通信テーブルは

send: $w_{h,s} \cap W_{h,r}$ recv: $W_{h,s} \cap w_{h,r}$

となる. 但しr 番目のPE に対してPE 番号s が持つ通信テーブルである. それを以下のように拡張する. 隣接PE 集合 $neibPE$ は $\{r \mid W_{h,s} \cap W_{h,r} \neq \phi\}$,

send: $W_{h,s} \cap W_{h,r}$ recv: $W_{h,s} \cap W_{h,r}$

この通信テーブルを使い, アグリゲートの交換をすれば各PE が $W_{h,k}$ の範囲のデータを持つようにP を分割できる.

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

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[1),4)] 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, 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. 本論文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論文の並列化の拡張は可能である. アグリゲート生成アルゴリズムの詳細は[1),4)] を参照する.

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. 例えば各PE がもつアグリゲートの行列は図3 のようになる.

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 アグリゲートのスムージング

式 (1) にしたがって, 行列 P を生成する. 問題行列に基づくグラフ上で, アグリゲートは1 層外側まで広

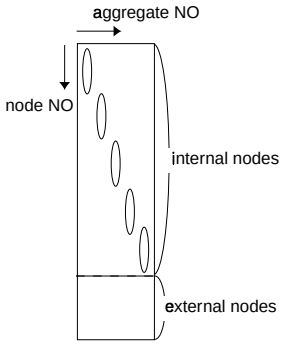


図 3 節点番号順にアグリゲートとなった場合の行列 $\tilde{P}$ の様子．図中の楕円は各アグリゲートを表す．楕円内の要素は 1 それ以外の要素は 0 となる．

がることになる．

問題行列 A_{level} の対角要素はオーバラップ領域についても正しい値を保持するように通信をしておく．すると担当節点集合内のみ非ゼロなベクトルについては通信を行うことなくヤコビ法 1 回で計算できる．

行列 P は各担当節点集合の 1 層外側（外部節点）まで正確な値を保持しなければならない．ここで拡張された通信テーブルを適用してアグリゲートの情報を通信する．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通信テーブルの RECV と同様の構造で，PE 番号ごとにアグリゲート番号を配列として持つ．P は図 4 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．

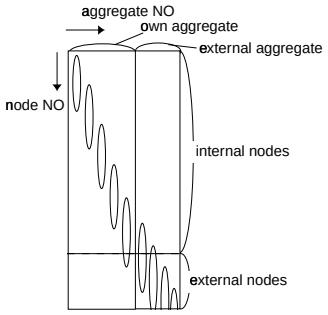


図 4 節点番号順にアグリゲートとなった場合の行列 P の様子．図中の楕円は各アグリゲートを表す．楕円内の要素は非ゼロ それ以外の要素は 0 となる．

4.3 行列積 P^TAP

この行列積は各担当領域内で独立に行なう．式 (4) 参照．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．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必要がある．

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．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し，外部アグリゲートと担当アグリゲートの要素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．

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．以上のステップで粗いレベル問題行列生成が完了する．

アルゴリズムの概略は図 5 のようになる．

```
/* 問題行列生成部 */
for level = 1 to LEVEL − 1 do
  clevel = level + 1
  Ã = filter(A_level)
  P̃_clevel = aggregation(Ã_level)
  P_clevel = smooth(Ã_level)
  /* 通信テーブルの拡張 */
  create_table(send, recv)
  /* 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_clevel, aggregate_table)
  A_clevel = P^T_clevel A_level P_clevel
  /* 外部アグリゲートの行を通信 */
  /* 外部アグリゲートテーブル更新 */
  sendrecv_A(A_clevel, aggregate_table)
  /* 粗いレベルの通信テーブル作成 */
  sendrecv_table(aggregate_table)
  /* 外部アグリゲートの対角要素の更新 */
  sendrecv_D(A_clevel)
end for
```

図 5 並列 問題行列生成部アルゴリズム

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 GeoFEM[7] ライブラリのソルバと同じインタフェースで実装し，評価する．

問題は 3 次元ポアソン方程式で以下のものである．

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

問題領域は XYZ 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 当たりの節点数を $50 \times 50 \times 50$ 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 法の反復計算部では 各レベルで対称ガウスザイデル法1 回を各PE 領域内で行ない, 領域間ではヤコビ法を行なう. 最も粗いレベルでは, 対称ガウスザイデル法を10 回行なう. 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 750MHz ×2, 1GB 主記憶) 128 ノードをMyrinet 2000 により接続したクラスタを用いた. ノード間通信はMPICH-GM により行なった.

表1 問題サイズと計算時間

PE 数	問題サイズ	反復回数	時間 (秒)
2PE	50×50×100	21	98.4
4PE	50×100×100	23	106.7
8PE	100×100×100	26	142.5
12PE	100×100×150	28	123.3
18PE	100×150×150	29	144.9
27PE	150×150×150	29	142.5
32PE	100×200×200	30	163.9
48PE	150×200×200	30	168.8
64PE	200×200×200	30	170.9

同じ領域分割でGeoFEM ライブラリに含まれているICCG 法[6] と本研究でのAMGCG 法とを比較する. 図6 は64PE の場合の各手法の残差減少の様子を表した. 図7 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6 では, AMGCG 法の残

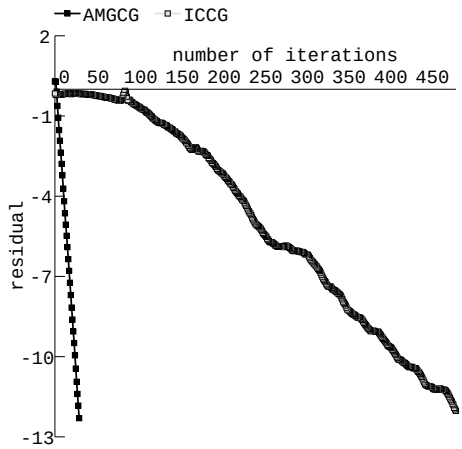


図6 残差の変化 :64PE,200x200x200 : 縦軸は相対残差 (2 ノルム) の対数, 横軸は反復回数

差は一定の割合で残差が減っている. この現象は, 誤差周波数成分の短波長成分から長波長成分までバランス良く減らしていく, というMG 法の特徴に基づいている.

図7 からAMGCG 法が大規模な問題においては

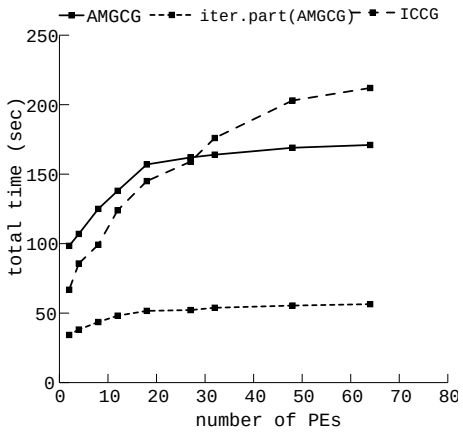


図7 AMGCG とICCG の実行時間 :iter.part (AMGCG) はAMGCG の 反復解法部にかかった時間

ICCG 法より有利であることがわかる.

また $Ax = b$ 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良い. 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. 行列積の計算と通信時間を計測した. 図8 では, 問題生成部の半分以上の時間はRAP の行

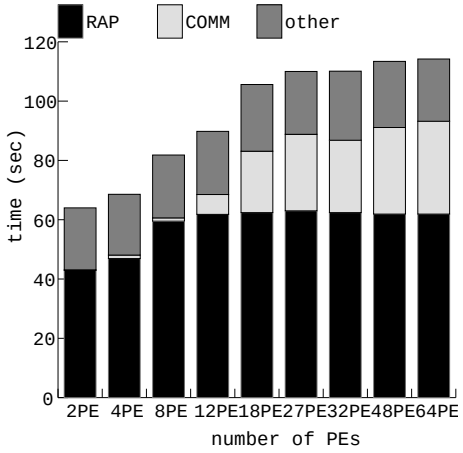


図8 問題生成部内の構成

列積に費やされている. 通信に関する時間は徐々に増えているが, ある程度の台数まで並列性が増すとそれ以降はあまり増加しない. 1 PE 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. これはPE 数を増やしても変わらないため, PE 数が $3 \times 3 \times 3$ より多くなってもあまり通信時間は変化しない.

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表にした. 但し全PE のうち, 割合が最小のものの値を用いた.

表 2 各レベルにおける担当節点の割合

PE 数	level1	level2	level3	level4
2PE	0.980	0.885	0.561	0.473
4PE	0.961	0.785	0.348	0.232
8PE	0.942	0.738	0.248	0.104
12PE	0.924	0.688	0.190	0.0615
18PE	0.906	0.640	0.138	0.0394
27PE	0.888	0.603	0.120	0.0309
32PE	0.906	0.640	0.134	0.0273
48PE	0.888	0.602	0.117	0.0185
64PE	0.888	0.601	0.112	0.0158

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, 通信の時間が増加していることが確認される.

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 法の並列化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$1/2 \sim 1/3$ 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.

一方で問題生成部の時間の内訳を見ると, 行列積 P^TAP と通信にかかる時間の合計は問題生成部の60% ～80% も占めている. 今後, 通信時間の隠蔽や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. 粗いレベルでの通信時間を少なくするようなデータ構造を別に考察する必要があるかも知れない.

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を考案することにより, 大規模な対称正定値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. 感謝致します.

参 考 文 献

1) Ray S.Tuminaro, and Chales Tong: "Parallel Smoothed Aggregation Multigrid : Aggregation Strategies on Massively Parallel",SuperComputing 2000 Proceedings, 2000
2) P.Vanek, M.Braezina, and J.Mandel:,"Convergence of Algebraic Multigrid Based on Smoothed Aggregation",Tech.Rep. report 126,UCD/CCM, Denver, CO, 1998
3) P.vanek: "Algebraic Multigrid Based on Smoothed Aggregation for Second and Fourth Order Problems", Computing,56(1996),pp. 179-196
4) Van Emden Henson, and Ulrike Meier Yang, "BoomerAMG: a Parallel Algebraic Multigrid Solver and Preconditioner", Lawrence Livermore National Laboratory technical report,UCRL-JC-141495
5) G. Haase, M. Kuhn, and S. Reitzinger: "Parallel AMG on Distributed Memory Computers", SIAM Journal on Scientific Computing, accepted (2002).
6) Kengo Nakajima, and Hiroshi Okuda: "Parallel Iterative Solvers with Localized ILU Preconditioning for Unstructured Grids on Workstation Clusters", IJCFD,1999,Vol. 12,pp.315-322
7) http://www.geofem.tokyo.rist.or.jp
8) Tony F. Chan, and Petr Vanek: "Multilevel algebraic Elliptic Solvers." UCLA Math Dept CAM Report 99-9, February 1999. Invited paper, in Proc. of High Performance Computing and Networking 1999, Amsterdam, April 1999, Lecture Notes in Comp. Sci. 1593, Springer, pp. 1001-1014.
9) Andrew J. Cleary, Robert D. Falgout, Van Emden Henson, Jim E. Jones, Thomas A. Manteuffel, Stephen F. McCormick, Gerald N. Miranda, John W. Ruge: " Robustness and Scalability of Algebraic Multigrid" SIAMJ 2000 on Scientific Computing Volume 21, Number 5 pp. 1886-1908
10) Andy Cleary, Rob Falgout, Van Emden Henson, and Jim Jones: "Coarse-Grid Selection for Parallel Algebraic Multigrid", Proceedings of the Fifth International Symposium on Solving Irregularly Structured Problems in Parallel, Springer-Verlag Lecture Notes in Comp. Sci.
11) Adams Mark .F: "A Parallel Maximal Independent Set Algorithm",Proceedings 5th copper mountain conference on iterative methods,1998