マルチグリッド前処理付き共役勾配法の並列化*

建部 修見　小柳 義夫†
東京大学理学部情報科学科‡

概要

マルチグリッド前処理付き共役勾配法 (MGCG 法) は, 共役勾配法の前処理としてマルチグリッド前処理を行なう方法であり, マルチグリッド法では収束しにくいような複雑な問題でも簡単に収束する. 本研究ではその MGCG 法の並列化を考え, 高並列で収束効率の高い MGCG 法を提案する. マルチグリッド法の並列化には様々な問題点が存在するが, MGCG 法の場合はそれらをうまく解決することが出来る. 次にその MGCG 法を富士通の並列計算機 AP1000 上に実装し, 最も収束の速い MGCG 法の考察を行なう. さらに並列計算機で良く使用されている Scaled CG 法との比較により評価を行なう.

ABSTRACT

A multigrid preconditioned conjugate gradient method (MGCG method), which uses the multigrid method as a preconditioner of the PCG method, converges rapidly even for the problems that the original multigrid method does not converge effectively. This paper considers parallelization of the MGCG method and proposes the MGCG method with high parallelism and high efficiency. There are various problems when the multigrid method is parallelized, however in the case of the MGCG method, these problems can be successfully settled. Next implementation of the MGCG method on the Fujitsu multicomputer AP1000 is performed and the most efficient MGCG method is studied. Then it is evaluated by comparing with the Scaled CG method that is often used on the multicomputers.

1　はじめに

マルチグリッド前処理付き共役勾配法 (MGCG 法) は, 共役勾配法の前処理にマルチグリッド法を用いた解法であり, 次のような性質を持っている. (1) 条件の悪い問題でも, 共役勾配法の反復回数が少ない. (2) 反復の回数がメッシュのサイズによらない. マルチグリッド前処理により, 問題の固有値分布は大部分が 1 の周りに集まり, わずかな数の固有値が 0 と 1 の間に散らばる. マルチグリッド法だけで解く場合は, このわずかな数の固有値が効率的な収束の邪魔となって, うまく収束してくれない. しかしながらマルチグリッド法を共役勾配法の前処理として使う時には, 特に問題となることなく収束する.

共役勾配法は, 反復中の演算が内積演算, 行列とベクトルの乗算, ベクトルの和, 差で構成されているため, ベクトル化, 並列化しやすい. またマルチグリッド法もスムージングを行なう方法次第で並列性がある. この研究では高い並列性を持ちかつ収束効率の高い MGCG 法を提案する. まず 2章では MGCG 法を簡単に説明し, 3章ではその MGCG 法の並列化について考察を行なう. この考察を踏まえ, 4章で富士通のマルチコンピュータ AP1000 上に実装し, その評価を行なう. 評価に当たって最も収束の速い MGCG 法を考えるとともに, 100% ベクトル化, 並列化可能で並列計算機上で良く用いられている Scaled CG 法 (SCG 法) との比較も行なった. この比較では MGCG 法は SCG 法に比べ計算時間が数十倍速かった.

---

*Parallelization of the Multigrid Preconditioned Conjugate Gradient Method

†Osamu TATEBE　Yoshio OYANAGI
E-mail: {tatebe,oyanagi}@is.s.u-tokyo.ac.jp
‡Department of Information Science, Faculty of Science, the University of Tokyo

2 MGCG 法

MGCG 法は前処理にマルチグリッド法を用いた共役勾配法である. 解くべき連立一次方程式を $L_l\boldsymbol{x}=\boldsymbol{f}$ とすると, MGCG 法の反復はプログラム 1のようになる. まず, 初期ベクトルを $\boldsymbol{x}^0$ と置く. この時, 初期残差は $\boldsymbol{r}^0=\boldsymbol{f}-L_l\boldsymbol{x}^0$ となる. そして, $L_l\tilde{\boldsymbol{r}}^0=\boldsymbol{r}^0$ をマルチグリッド法を使って近似的に解き, 初期方向ベクトルを $\boldsymbol{p}^0=\tilde{\boldsymbol{r}}^0$ とする. そしてプログラム 1のループを収束するまで繰り返す.

$i=0$;
while (!convergence) {
  $\alpha_i=(\tilde{\boldsymbol{r}}_i,\boldsymbol{r}_i)/(\boldsymbol{p}_i,L_l\boldsymbol{p}_i)$;
  $\boldsymbol{x}_{i+1}=\boldsymbol{x}_i+\alpha_i\boldsymbol{p}_i$;
  $\boldsymbol{r}_{i+1}=\boldsymbol{r}_i-\alpha_iL_l\boldsymbol{p}_i$;
  convergence test;
  Relax $L_l\tilde{\boldsymbol{r}}_{i+1}=\boldsymbol{r}_{i+1}$ using the Multigrid method ——————— (A)
  $\beta_i=(\tilde{\boldsymbol{r}}_{i+1},\boldsymbol{r}_{i+1})/(\tilde{\boldsymbol{r}}_i,\boldsymbol{r}_i)$;
  $\boldsymbol{p}_{i+1}=\tilde{\boldsymbol{r}}_{i+1}+\beta_i\boldsymbol{p}_i$;
  i++;
}

プログラム 1. MGCG 法の反復

このプログラムの中で, (A) の部分がマルチグリッド前処理の部分である. またマルチグリッド法はグリッドの列を $\{L_i\}$ とすると, プログラム 2の様に再帰的に表すことが出来る.

Vector MG(L_l, $\boldsymbol{f}$, $\boldsymbol{x}$, γ, μ)
{
  if (l == coarsest_level) Solve $L_l\boldsymbol{x}=\boldsymbol{f}$;
  else {
    $\boldsymbol{x}$ = pre_smoothing(L_l, $\boldsymbol{f}$, $\boldsymbol{x}$, μ);
    $\boldsymbol{d}$ = restrict($\boldsymbol{f}-L_l\boldsymbol{x}$);
    $\boldsymbol{\nu}$ = initial_$\boldsymbol{x}$;
    repeat (γ) $\boldsymbol{\nu}$ = MG(L_{l-1}, $\boldsymbol{d}$, $\boldsymbol{\nu}$, γ, μ);
    $\boldsymbol{x}$ = $\boldsymbol{x}$ + prolongate($\boldsymbol{\nu}$);
    $\boldsymbol{x}$ = post_smoothing(L_l, $\boldsymbol{f}$, $\boldsymbol{x}$, μ);
  }
  return $\boldsymbol{x}$;
}

プログラム 2. マルチグリッド法

このプログラム中で, restrict とはグリッドレベル i から $i-1$ への変換を行なう操作を表し, prolongate とはその逆のグリッドレベル $i-1$ から i への変換を行なう操作を表している. γ が 1 の時のマルチグリッド法を特に V-cycle マルチグリッド法, 2 の時を W-cycle マルチグリッド法と呼ぶ (図 1). このマルチグリッド法が共役勾配法の前処理としての数学的に妥当であるための条件は, 既に [10] で調べてあり, スムージング法としてレッドブラック対称ガウスザイデル法, マルチカラー SSOR 法, ADI 法などの方法を前処理に使い, プレスムージングとポストスムージングで同じ方法を使えば前処理としての条件は満たされる.

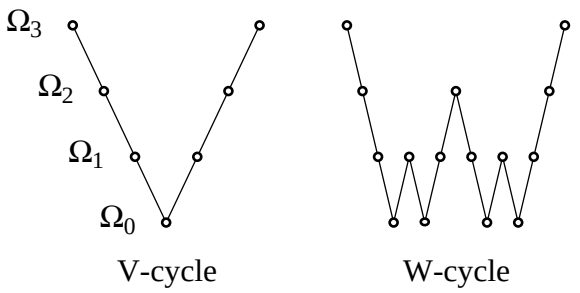

図 1. マルチグリッド法のサイクル

3 MGCG 法の並列化

この章では MGCG 法のマルチコンピュータ上への実装を目的とした並列化を考える. MGCG 法は共役勾配法の前処理としてマルチグリッド法を使うものであるので, 共役勾配法とマルチグリッド法の両方の方法を並列化しなければならない. 共役勾配法が並列化しやすいことは既に述べたので, まずマルチグリッド法を並列化することを考える.

3.1 マルチグリッド法の並列化

マルチグリッド法を並列化する場合は次に様な問題が出てくる.

1. 高い並列性を持ったスムージングを行なう必要がある.
2. 粗いグリッドになればなるほど通信のオーバーヘッドが見えてくる
3. 粗いグリッドではプロセッサの使用率が低くなり, 遊んでいるプロセッサが出てしまう

マルチグリッド法は，残差の減少率が最も粗いグリッドによるので，有効な限り粗いグリッドまで使った方が収束が速くなる．しかしながら問題が複雑で条件の悪い場合は，ただ粗いグリッドまでいけば収束が加速される訳ではなく，余り並列性の高くないより強力なスムージング (ADI 法や不完全 LU 分解など) を行なわなければならない．従って，マルチグリッド法をマルチコンピュータ上に実装する時は，上の 1 番目の問題をクリアすることは難しく，計算機の持つ並列性を存分に発揮できない．また 2 番目，3 番目の問題点を解決するために，逐次型と同じアルゴリズムであることをあきらめ，グリッドが粗くなった時もそれぞれのプロセッサで計算するデータが減らないような並列マルチグリッド法を考えている．

3.2 MGCG 法の並列化

MGCG 法をマルチコンピュータ上に実装する時，そのまま実装しようとすると前節で考えたマルチグリッド法を並列化する時に出てくる問題と同様の問題が出てくる．しかしながら MGCG 法の場合は，マルチグリッド法はあくまで共役勾配法の前処理として使うため，複雑な問題を解くときに並列性の高くないより強力なスムージングを使うことは必要ではない．このことは次の固有値分解で議論する．従ってマルチグリッド前処理のスムージングは並列性が高く，反復解法としても優れていて，なおかつ前処理として数学的に妥当な解法であるレッドブラック対称 SOR 法 (RB-SSOR 法) を使うことにする．

この時，次に考えることはどれだけグリッドを粗くするのがいいかということである． MGCG 法の場合はマルチグリッド法と違い複雑な問題でもグリッドを粗くすればするほど収束は加速される．従って収束を加速するために，出来る限り粗いグリッドまで使いたい． ここで問題となるのは 3.1節の 2 番目と 3 番目の問題である．この部分は計算機の性能がでる範囲まで粗いグリッドを使うことで対処する．つまり並列計算機の計算 / 通信性能と収束の速さという点でのトレードオフということになる．ここでのトレードオフは，マルチグリッド前処理で使うグリッドの数，スキーム，RB-SSOR 法の反復回数と並列計算機の計算 / 通信性能の間のものということになる．次節では固有値解析を行ないこのようにグリッドの数を減らした時，並列性のあるスムージングを行なった時のマルチグリッド前処理について考察する．

3.3 固有値解析

この節では複雑で条件の悪い問題に対し，最も粗いグリッドで厳密に解かず，また使うグリッドの数を減らし，しかも高並列なスムージング法を使うマルチグリッド前処理を行なった時の行列の固有値の分布を調べる．共役勾配法の反復回数は初期値，係数行列の固有値分布，右辺で決まるが，その中で係数行列の固有値分布は特に重要である．反復回数は初期残差に含まれる係数行列の固有ベクトルの数で決まるので，固有値解析を行なうことにより初期値，右辺が最悪の場合の反復回数を知ることが出来る．設定した問題は次の様なものである． 2 次元のディリクレ境界条件付きポアソン方程式

$$-\nabla\left(k\nabla u\right) = f \quad \text{in} \quad \Omega = [0,1]\times[0,1]$$

$$\text{with} \quad u = g \quad \text{on} \quad \partial\Omega,$$

で，物理定数 k は図 2のように不均一で非対称に定める． 図 3は，この問題を有限要素法で 16×16 に離散化し，出てきた連立一次方程式を次のようなマルチグリッド前処理を行なった後の行列の固有値分布である．

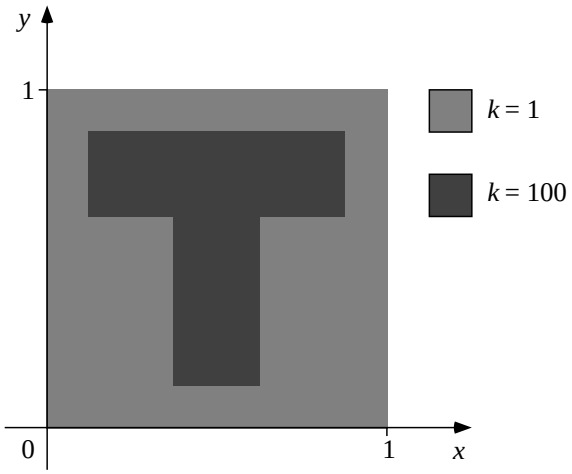

図 2. 問題の物理定数

ここで使ったマルチグリッド前処理は，グリッドを 2 つしか使わないで，粗い方のグリッドで解く時に，スムージングで使う緩和計算を行なっただけのマルチグリッド法である． この図には比較のため ICCG(1, 2) の前処理を行なった後の行列の固有値分布も付けてある． この固有値分布をみれば分かるように，マルチグリッド前処理の方はかなりの固有値が 1 の周りに縮重していて，固有値の分布も片

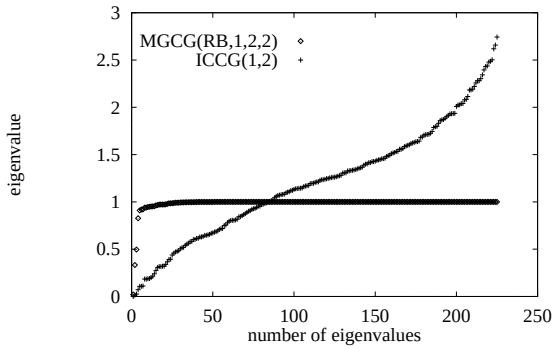

図 3. 前処理後の固有値分布

寄っている．また最小固有値は ICCG 法に比べ引き上げられている．いくつかの固有値は分散しているが，これらは共役勾配法にとっては何の問題もなく，それらは実際に共役勾配法のそれぞれ一反復でなくなってしまう． 前処理を行なった問題の固有値がこれらの性質を持っているため，グリッドを一つ少なくしても，またスムージングとして高並列な RB-SSOR 法を用いても反復回数はそんなに多くならないと考えられる．

この例は，3 つのグリッドを使うところを 2 つにして，しかも粗いほうのグリッド上では厳密に解かない場合であったが，共役勾配法の前処理として使う限り，極端に収束性が悪くなることはなく，また ICCG 法に比べると非常に良い前処理といえる．

3.4 領域分割

次に，マルチコンピュータをターゲットとした並列化を行なうためのデータの領域分割について考察する． 行列とベクトルの乗算，マルチグリッド法のリストリクション，スムージング法のそれぞれは全て近接通信を必要とする．従ってなるべく連続的になるようにプロセッサにデータを割り付けるのがよい．以下では領域は 2 次元と仮定する． このときプロセッサが M 台あるとすると，(A) ベクトルを領域の 1 方向だけで分割し，プロセッサを論理的に $M \times 1$ に割り当てる方法と，(B) 2 方向で分割し，プロセッサを $\sqrt{M} \times \sqrt{M}$ に割り当てる方法が考えられる (図 4)．

問題の大きさを N とした場合，これらの方法についてそれぞれの通信量，計算量を比較してみる．ターゲットとする並列計算機が 2 次元トーラスメッシュで隣接通信がそれぞれ同時にできるとすると，計算量はいずれも $O(N/M)$ であるが，通信量は (A) は $O(\sqrt{N})$ で，(B) は $O(\sqrt{N}/\sqrt{M})$ である． これによると 2 方向で分割した場合のほうが，1 方向で分割した場合に比べ通信量が $1/\sqrt{M}$ 倍となっているのが分かる．しかしながら (B) の場合は行列とベクトルの乗算，スムージング法で 4 方向のプロセッサ間で通信を必要とし，レストリクションではレストリクションの方法によって違ってくるが普通 6 から 8 方向のプロセッサ間で通信が必要になる． (A) の場合はいずれ場合でも 2 方向で良い． これらのことを考えると，問題の大きさが一定の場合はプロセッサの台数が多くなれば 2 方向でデータを分割したほうが有利ということになる．

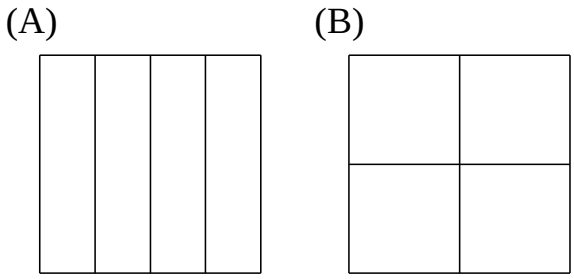

図 4. データの割り付け方法

また (A)，(B) いずれの場合でもそれぞれのプロセッサに割り付けるデータは，マルチグリッド法のレストリクション，プロロンゲーションという操作を均一に行なうためには，領域の境界にあるデータはそれを境界とするプロセッサが全て同じデータを重複して持つ必要がある．

4 AP1000 への実装

4.1 データ割り付け

まずデータの割り付けの仕方による違いを見る．前節の (A)， (B) それぞれのデータ分割の仕方で AP1000 の上に実装して時間を測定した結果が，表 1，2である． またこの表を元にスピードアップのグラフを作ると，図 5, 6のようになる．

この時使用した問題は 3.3節の固有値解析で使用した問題について，k の値を一定，ソース項 f は 0 で境界条件は $y = 1$ でのみ値を持つとした問題である． 使った MGCG 法のマルチグリッド前処理はグリッドを 2 つしか使わないもので，スムージング法はレッドブラックガウスザイデル法の 2 反復を使用した．

size	# iter.	1×1 (台)	2×1	4×1	8×1	16×1	32×1	64×1
63^2	19	6.72	2.93	1.65	0.85	0.55	-	-
127^2	38	62.48	29.46	14.41	6.73	3.94	2.21	-
255^2	71	476.99	236.95	114.84	58.20	29.24	15.43	9.40

(sec)

表 1. (A) の場合 {MGCG(RB, 1, 2, 2)}

size	# iter.	1×1 (台)	2×2	4×4	8×8
63^2	19	7.09	1.66	0.50	0.23
127^2	38	60.67	14.26	3.46	1.01
255^2	71	455.4	113.8	27.3	6.49

(sec)

表 2. (B) の場合 {MGCG(RB, 1, 2, 2)}

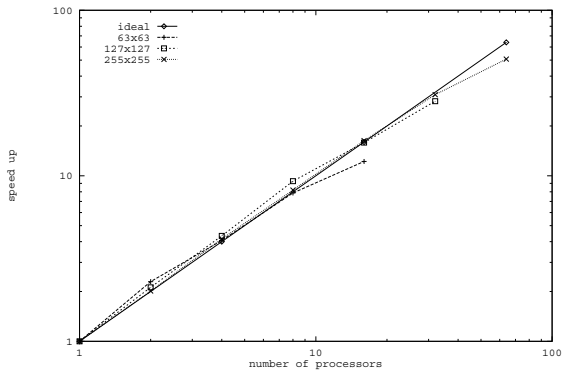

図 5. (A) 1 方向で分割した場合

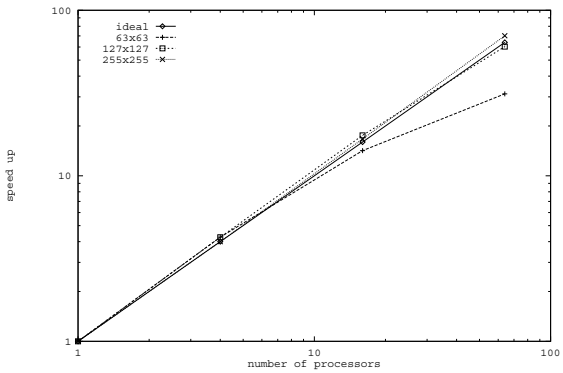

図 6. (B) 2 方向で分割した場合

表中で - となっているのは, 1 プロセッサにデータが実質的に 1 つずつまたは遊んでいるプロセッサが出てしまい, 明らかに他のデータと同列に並べても余り意味がないと思われるところである.

これらの表, 図を見ても分かるが, 問題の大きさを一定にすると, プロセッサの台数が多くなった時に, (B) の 2 方向で分割した場合の方が良い結果となっている. また, いずれの方法もスーパーリニアとなっているが, これは問題の大きさを固定し, プロセッサの台数を増やしていったため, 1 プロセッサあたりのデータ量が少なくなり, キャッシュのヒット率, ラインセンドのヒット率が上がった影響である. (B) の場合のついてキャッシュのヒット率を計測すると図 7のようであった.

この図 7を見ると, 特にメッシュサイズが 256×256 の時はキャッシュのヒット率が著しく変わっていることが分かる. 以後はこの結果を踏まえ, データのプロセッサへの割り当ては (B) 方式, つまり領域を 2 方向で分割しプロセッサに割り当てる方式で行なうことにする.

4.2 グリッドの数

次に前処理を行なう V-cycle マルチグリッド法のグリッドの数を変えた時の, 収束回数, 計算時間を測定した. 問題は固有値解析を行なったものと同じ問題で, ソース項 f として図 8を使い, 境界条件は全て 0 とした.

この問題において, グリッドの数を変えていった

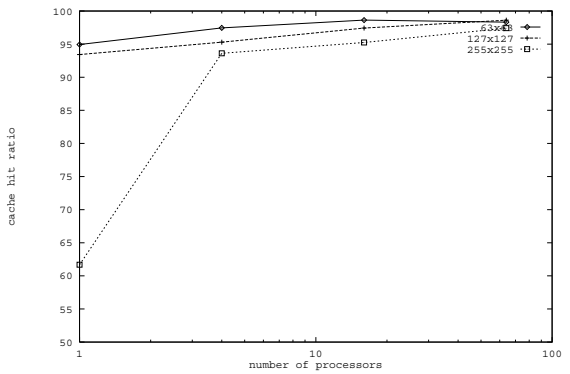

図 7. キャッシュのヒット率

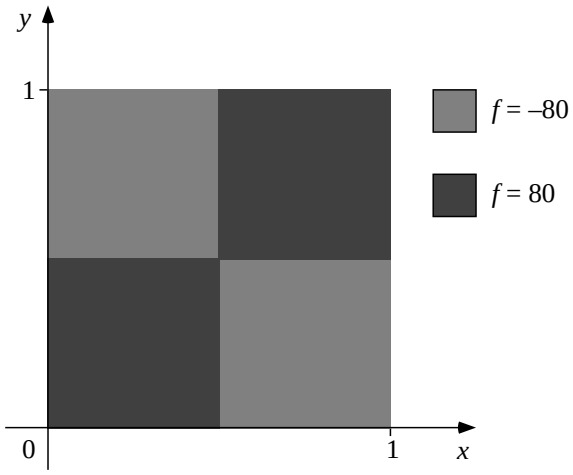

図 8. 問題のソース項

時の反復回数, 計算時間は表 3, 4である. 表 3は問題を 256×256 に離散化して解いたもので, 表 4は問題を 512×512 に離散化した解いたものである. この実験ではプロセッサは 64 台使用している.

表中の coarsest というのは一番粗いグリッドにおけるメッシュの数である. この欄が 8×8 というところでは, 1 プロセッサは実質的には一つのデータしか受け持っていない. また per 1 iter. というのは共役勾配法の 1 反復にかかった時間である. この実験結果によると, グリッドをたくさん使えば使うほど収束が速くなっている. この点がオリジナルのマルチグリッド法だけで解く場合との違いである.

グリッドを粗くしていった時には扱うデータは 2 次元の問題の場合 1/4 に減る. 一つ粗いグリッドを付け加え, 使うグリッドの数を増やした時には, それまで一番粗いところで扱っていたデータの 1/4 の

grids	iter	time (s)	per 1 iter.	coarsest
2	135	11.35	0.084	128×128
3	64	6.37	0.100	64×64
4	32	3.43	0.107	32×32
5	17	1.91	0.112	16×16
6	10	1.18	0.118	8×8

表 3. グリッドの数と計算時間の関係 (256×256)

grids	iter	time (s)	per 1	coarsest
2	270	104.4	0.387	256×256
3	129	56.1	0.435	128×128
4	64	29.0	0.453	64×64
5	33	15.3	0.463	32×32
6	17	8.00	0.471	16×16
7	11	5.24	0.476	8×8

表 4. グリッドの数と計算時間の関係 (512×512)

データを新たに処理することになる. 従ってどんどん粗いグリッドを付け加えていっても, 増える演算回数はどんどん少なくなる. 処理するデータの量が少なくなるので, キャッシュのヒット率, ラインセンドのヒット率は向上し, 粗いグリッドを増やしても一反復当りの時間が余り増えないことになる. このことについては次節でもう少し細かく議論する. この結果から見ると, むしろグリッドを一つ増やすとほぼ反復回数が半分になるということが, 計算時間に効いている. 二つの実験データを比べると, もっとも粗いグリッドで厳密に解かない場合, マルチグリッド法の残差の減少率はもっとも粗いグリッドのメッシュ間隔によるということが良く分かる.

4.3 スムージング法

次にスムージング法の反復回数を変えた時の計算時間, 反復回数を調べる. 使用した問題は前節と同じ問題である. その結果は表 5, 6である.

これらの表 5, 6によると, それぞれのグリッドの数についてスムージング法の反復回数は 1 回ないしは 2 回行なうのが最も速く収束している. スムージング法の反復回数を 1 回増やすと始めは反復回数は 70% 程少なくなっているが, どんどん増やしていくと反復回数の減り方が鈍くなっている. 一方, グリッドを 1 つ増やすとほぼコンスタントに反復回数は半分になっている. また多くのグリッドを使う場合

grids	1		2		3		4		5		6	
smoothing	iter	time	iter	time	iter	time	iter	time	iter	time	iter	time
1	486	14.55	184	11.55	87	6.43	44	3.49	22	1.83	13	1.13
2	344	13.51	135	11.35	64	6.37	32	3.43	17	1.91	10	1.18
3	281	13.65	111	11.69	53	6.63	27	3.65	14	1.99	9	1.34
4	243	14.06	97	12.28	47	7.09	24	3.91	12	2.06	8	1.43
5	217	14.57	87	12.86	42	7.41	21	4.01	11	2.21	8	1.68
6	198	15.13	80	13.52	38	7.68	20	4.37	10	2.30	7	1.69

表 5. スムージング法の反復回数と計算時間 (256×256)

grids	5		6		7	
smooth.	iter	time	iter	time	iter	time
1	44	15.14	23	8.03	13	4.61
2	33	15.27	17	8.00	11	5.24
3	27	15.70	14	8.27	9	5.39
4	24	16.80	12	8.54	8	5.77
5	21	17.19	11	9.15	8	6.75
6	20	18.74	10	9.53	8	7.71

表 6. スムージング法の反復回数と計算時間 (512×512)

はスムージング法の反復回数を増やすと反復 1 回にかかる時間が大幅に増えてしまうのに対し，グリッドを増やす方は，増やしていくにつれ反復 1 回にかかる時間の増え方がどんどん減っているのが分かる．従ってグリッドを余り使わない場合は，スムージング法の反復回数が少ない時に，スムージング法の反復回数を増やして計算時間が短くなることがあるが，グリッドを多く使う時はスムージング法の反復を 1 回しかしないのが最も速く収束している．

4.4 キャッシュのヒット率と台数効果

次にプロセッサ数を変化させた時のキャッシュのヒット率を調べた． 問題は前節と同じ問題を 256×256 に離散化したもので，解法は最も収束の速かった，マルチグリッド前処理に 6 つのグリッドを使い，スムージング法の反復回数を 1 回とした MGCG 法である．この場合は先ほど 4.1で見たヒット率に比べ粗いグリッドをたくさん使用しているため，通信遅延が隠せない，反復回数が少ないので逐次部分の影響が出るなどの理由で同じようにスーパーリニアとなるような台数効果は得られないと考えられる．結果は表 7である．

# procs.	time (s)	hit ratio	speed up
1	71.80	93.57	1
4	17.99	94.07	3.99
16	4.53	95.49	15.85
64	1.13	97.96	63.54

表 7. キャッシュのヒット率

しかしながらこの結果を見ると，プロセッサが多くなるとキャッシュのヒット率がかなり改善されているため，スピードアップはほぼリニアとなっている． やはり，多くのプロセッサを使い巨大なデータを分割することによるキャッシュのヒット率の向上という効果は素晴らしいものである．

4.5 SCG 法との比較

最後に 100% ベクトル化，並列化可能でしかも収束が速いといわれている SCG 法 [5] との比較を行なった．SCG 法というのは前処理として対角スケーリングを行なう共役勾配法である．SCG 法は ICCG 法などに比べ前処理が簡単なため反復回数は増えるが，共役勾配法の反復が 100% 並列化可能なため 1 反復に要する時間が短く，全体では計算時間は短縮されるといわれている方法である．

MGCG 法で数値実験を行なった問題を同じ条件で SCG 法で解いた結果が表 8である． ただしこのデータを出すに当たって， データの分割方法は MGCG 法と同様に分割し，境界上のデータはそれぞれのプロセッサで重複して持っている．

この結果より，SCG 法の 1 反復にかかる時間は今まで考察してきた MGCG 法において前処理を除いた時間にほぼ等しくなっていることが分かる．し

size	iter.	time (s)	per 1 iter.
256^2	972	15.87	0.0163
512^2	1954	165.71	0.0848

表 8. SCG 法の計算時間

かしながら収束までの反復回数はそれぞれ 75 倍, 150 倍もかかっているため, SCG 法の計算時間は MGCG 法と比べ実に 14 倍から 36 倍遅くなっている.

5 まとめ

この研究では, 対称正定値である大規模疎行列の解法として有望な MGCG 法のマルチコンピュータ上への並列化を考察した. 複雑な問題でマルチグリッド法を効率良く収束させる為には余り並列性のない強力なスムージング法を用いなければならないので, マルチグリッド法を並列化する場合は様々な問題が生じてくる. しかしながら MGCG 法の場合は, 複雑な問題に対して前処理のマルチグリッド法に高並列なスムージング法を使っても収束効率は悪くならない. さらにそのような高並列なスムージング法を使っても, 粗いグリッドを使えば使うだけ収束が速くなる. 従って MGCG 法を並列化する時は, その並列計算機のパワーを十分に発揮できる粗さのグリッドまで使用すれば, MGCG 法は高並列でかつ非常に収束効率の高い解法となることを示した. また実際にその MGCG 法の台数効果を調べるとキャッシュのヒット率の増大という効果もあり, ほぼリニアな効果が得られていた. さらに並列計算機上で有望視されている SCG 法とこの MGCG 法との比較を行い, MGCG 法が SCG 法の数十倍速く収束することを示した.

謝辞

本研究を行なうにあたり富士通並列処理研究センターの並列計算機 AP1000 を使用した. ここに深謝する. なお本研究は科学研究費補助金重点領域研究 (課題番号 04235203) の援助を受けた.

参考文献

[1] Adams, L. M., *Iterative Algorithms for Large Sparse Linear System on Parallel Computers.* PhD thesis, University of Virginia, November 1982.

[2] Braess, D., "On the Combination of the Multigrid Method and Conjugate Gradients," in *Multigrid Methods II* (W. Hackbusch and U. Trottenberg, eds.), vol. 1228 of *Lecture Notes in Mathematics*, pp. 52–64, Springer-Verlag, 1986.

[3] Hackbusch, W., "Multi-grid Convergence Theory," in *Multigrid Methods* (W. Hackbusch and U. Trottenberg, eds.), vol. 960 of *Lecture Notes in Mathematics*, pp. 177–219, Springer-Verlag, 1982.

[4] Hackbusch, W., *Multi-Grid Methods and Applications.* Springer-Verlag, 1985.

[5] 速水 謙, 原田 紀夫, "ベクトル計算機における Scaled CG 法の有効性について," 情報処理学会, 数値解析研究報告, 17–4, 1986 年 7 月.

[6] Maitre, J. F. and F. Musy, "Multigrid methods: Convergence theory in a variational framework," *SIAM J. Numer. Anal.*, vol. 21, pp. 657–671, 1984.

[7] Modi, J. J., *Parallel Algorithms and Matrix Computation.* Oxford University Press, 1988.

[8] Ortega, J. M., *Introduction to Parallel and Vector Solution of Linear Systems.* Plenum Press, 1988.

[9] Stüben, K. and U. Trottenberg, "Multigrid methods: Fundamental algorithms, model problem analysis and applications," in *Multigrid Methods, Proceedings of the Conference Held at Köln-Porz* (W. Hackbusch and U. Trottenberg, eds.), vol. 960 of *Lecture Notes in Mathematics*, pp. 1–176, Springer-Verlag, 1982.

[10] 建部 修見, 小柳 義夫, "マルチグリッド前処理付き共役勾配法," SWoPP'92, 情報処理学会, 数値解析研究報告, No. 42, pp. 9–16, 1992 年 8 月.

[11] Wesseling, P., "Theoretical and practical aspects of a multigrid method," *SIAM Journal on Scientific and Statistical Computing*, vol. 3, pp. 387–407, 1982.