ハイパフォーマンス
コンピューティング 52−10
（1994. 7. 23）

マルチグリッド前処理付き自乗共役勾配法の並列化

襲田 勉 小柳義夫
東京大学理学部情報科学科

概要

自乗共役勾配(CGS)法の前処理として修正不完全LU分解(MILU)が良く使われているが、この前処理は収束率の改善が不十分であり、しかも並列性が低く並列計算機上での効率的な実装が難しい。この論文では非対称係数行列の問題に対して、マルチグリッド前処理付き自乗共役勾配(MGCGS)法を適用し、実際に並列計算機上で実装することにによってその有効性を示す。マルチグリッド(MG)法は多種類の格子点を使うことで収束を加速させる方法であるが、並列化に際しては様々な問題点を生じる。MGCGS法ではこれらの問題点が解決されることを示し、並列計算機上で効率の良いMGCGS法を考察する。さらにMILUCGS法との比較を行ないMGCGS法が優れた方法であることを示す。

Parallelization of the Multigrid Preconditioned Conjugate Gradient Squared Method

Tsutomu Osoda and Yoshio Oyanagi
Department of Information Science, Faculty of Science, the University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113, JAPAN

Abstract

The modified incomplete LU decomposition(MILU) preconditioner, which is usually applied to the conjugate gradient squared method, does not improve convergence rate sufficiently and cannot be implemented efficiently on parallel computers. We apply the mutitigrid preconditioned conjugate gradient squared(MGCGS) method to a linear equation with unsymmetric coefficient matrix and implement this method on parallel machines. This paper shows the efficiency of the MGCGS method. The multigrid method shows the rapid convergence exploiting grids of difference sizes but produces many problems in case of parallel implementation. This paper presents the solution of the MGCGS method to these problems and the efficient implementation of the MGCGS method on parallel machines. Moreover this paper presents that the MGCGS method is also superior to the MILUCGS method on parallel machines.

1 問題

この論文ではマルチグリッド前処理付き自乗共役勾配 (MGCGS) 法 ([6]) を並列計算機上に実装し、並列計算機上で効率的なマルチグリッド前処理 (MG) を考える。

1.1 対象とする方程式

以下の議論では次のような定常状態における二次元の移流拡散方程式を対象とする。また境界条件は固定境界条件とする。

$$\mathrm{div}\Big(-k\nabla u + \boldsymbol{b}u\Big) = f(x,y) \text{ in } \Omega \subset \Re^2 \text{ with } u = 0 \text{ on } \Gamma \tag{1}$$

ここで Γ は領域 Ω の境界である。物理定数は以下のように定める。拡散係数は図 1のように領域内で二種類の異なった値を持つように定める。移流ベクトルの向きは図 2のように y 軸方向であるとする。その移流ベクトルの大きさは V_0 をある適当な正の値として、次の式により定義する。

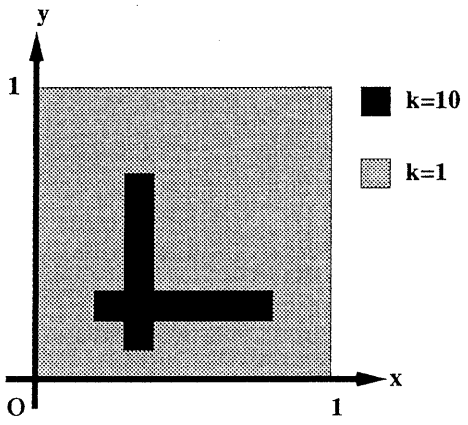


図 1. 拡散係数

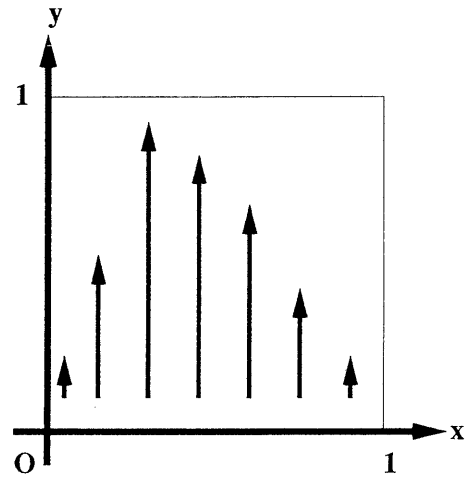


図 2. 移流ベクトル

$$\|\boldsymbol{b}\| = -9V_0 x(x - \frac{2}{3}) \text{ when } 0 \le x \le \frac{1}{3} \tag{2}$$

$$\|\boldsymbol{b}\| = -\frac{9}{4}V_0(x-1)(x + \frac{1}{3}) \text{ when } \frac{1}{3} \le x \le 1 \tag{3}$$

ソース項は図 3のように定義する。

1.2 差分スキーム

MG 法では格子点間隔の減少のために、粗い格子点上で離散化した方程式の性質が悪化することがあるということが知られている。方程式を粗い格子点上で離散化し直す場合には、中心差分法を使って離散化し直すと方程式の性質が悪化し、風上差分法を使って離散化し直すと方程式の性質の悪化は防げるものの中心差分法を使った場合に比べて収束性が悪化する。そこでここでは、最も細かい格子点上では中心差分法を使って離散化し、粗い格子点上では風上差分法を用いて離散化することにする。このことについては後で詳しく調べる。

2 並列化

2.1 領域分割とデータの割り付け

並列計算機上で並列に実装するための領域の分割について見てみる。問題の大きさを N、プロセッサー台数を M とする。2 次元の領域を分割してデータをプロセッサーに割り当てる方法には、領域を 1 方向にのみ分割しプロセッサーを論理的に $M \times 1$ に割り当てる方法と、図 4のように 2 方向で分割しプロセッサーを $\sqrt{M} \times \sqrt{M}$ に割り当てる方法がある。ここでは後者のような割り当て方を行なう。その場合行列とベクトルの乗算、スムージングでは隣接する 4 方向のプロセッサー間で通信を必要とし、レストリクションではそれに対角線方向も含めた 8 方向のプロセッサーと通信する必要がある。各プロセッサーにおける計算量 $O(\frac{N}{M})$ で、通信量は $O(\frac{\sqrt{N}}{\sqrt{M}})$ となる。

MG 法の場合には、レストリクションやプロロンゲーションなどの操作を均一に行なうために、領域の境界にあるデータをそれを境界とするプロセッサーが全て同じデータを重複して持っている。

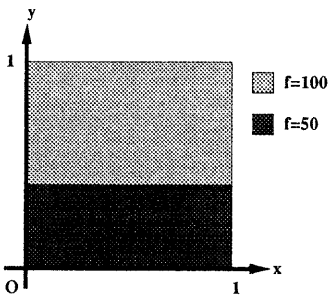


図 3. ソース項

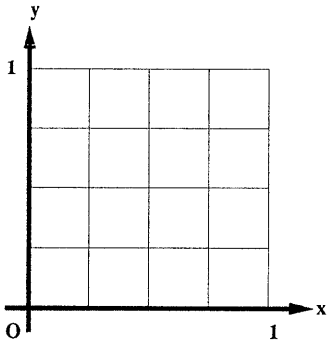


図 4. 領域分割

2.2 MG 法と MGCGS 法

MG 法を並列計算機上に効率良く実装するためには、並列性の高いスムージング (Red-Black Gauss Seidel(RBGS) 法など) が必要になる。通常は中心差分法など単一の差分法を用いるために、粗い格子点上になるほど格子点間隔の減少のために係数行列の非対称性が増加し、強力なスムージング (ADI 法、 ILU 法など) を用いなければならない。しかしながらこれらの方法を並列計算機上で効率良く実装することは困難である。そこで前述のように最も細かい格子点上では中心差分法を使い、粗い格子点上では風上差分法を使うことにする。そうすることによって粗い格子点上での係数行列の非対称性の増加が防げ、スムージングに並列性の高い方法を用いることができるようになるが、 MG 法ではそれが原因で中心差分法のみを用いたときよりも収束性が悪化してしまう。ところが MGCGS 法では中心差分法のみの場合と比較して、ほとんど収束性の悪化が見られない。図 5、図 6はそのことを示している。横軸は係数行列の非対称性の度合を縦軸は収束までの反復回数を示している。図の中で single とあるのが中心差分法だけを、 double とあるのが中心、風上差分法を併用した場合である。

そのような離散化法を考えた場合、 MG 法の残差の減少率は最も粗い格子点に依存しているので逐次計算機上ではなるべく粗い格子点まで使う方が良い。しかしながら並列計算機上では粗い格子点上になるほど通信のオーバーヘッドが顕著になり、また粗い格子点では idle のプロセッサーが生じる。したがって並列計算機上で効率の良い MGCGS 法とは機械の通信性能、計算性能、収束性とのトレードオフによって決まる。そのことはこの後の数値実験で詳しく考察する。

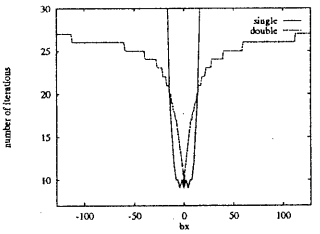


図 5. MG 法

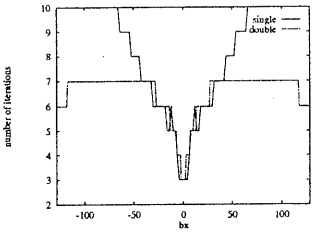


図 6. MGCGS 法

3 数値実験と考察

この章では数値実験により効率の良い MG 前処理を考察する。実験で用いた計算機は富士通の AP-1000 である。

3.1 グリッドの種類数

この章では MG 法における格子点の種類数を変えていったときの収束までの反復回数、計算時間を測定した。MG 前処理には V サイクルでスムージングに RBGS 法 1 反復を用いたものを使っている。この数値実験では 64 台のプロセッサーを使用している。実験の結果は表 1に示されている。表 1で最も粗い格子点数というのは MG 法でどこまで粗い格子点を使うかということを示している。ここで 7×7 というのは 1 プロセッサー当たり 1 個のデータを処理していることになる。

表 1. グリッド数と収束性

最も粗い格子点	反復回数	計算時間 (秒)	1 反復あたりの計算時間
1×1	8	1.58	0.198
7×7	12	1.79	0.149
15×15	19	2.46	0.129
31×31	41	3.95	0.0963
63×63	81	6.02	0.0743
127×127	213	7.05	0.0331

表 1の計算時間から MGCGS 法ではなるべく粗い格子点まで用いた方が計算時間が短いように考えられる。またこれは反復回数にも現れていて、1 種類の粗い格子点を用いるたびにほぼ反復回数が $\frac{1}{2}$ になっている。1 種類の粗い格子点を用いるたびに、2 次元の場合には扱うデータの増加が $\frac{1}{4}$ に減ることを考えれば粗い格子点を用いるほど収束性が改善されているように思える。しかしながら 1 反復あたりの計算時間から考えると 1×1 まで粗くした場合に急激に 1 反復あたりの計算時間が増加していることがわかる。これは粗い格子点になればなるほど idle のプロセッサーが生じ、通信時間のオーバーヘッドが大きくなるためである。このため通信時間が計算時間に比べて非常に遅い計算機では 1×1 まで粗くした場合には、かえって計算時間がかかることが考えられる。

3.2 スムージングの回数

この章では格子点の種類数とスムージングの回数を変化させたときの収束までの反復回数と計算時間を測定した。やはりMG前処理にはVサイクルでスムージングにRBGS法を用いたものを使っている。この数値実験では64台のプロセッサーを使用している。実験の結果は表2に示されている。表2からスムージングの反復

表2. スムージングと収束性

スムージングの回数	1		2		3		4		5		6	
最も粗い格子点	iter	time	iter	time	iter	time	iter	time	iter	time	iter	time
1×1	8	1.58	6	1.83	5	2.04	5	2.53	4	2.47	4	2.87
7×7	12	1.79	7	1.50	5	1.45	5	1.80	5	2.15	4	2.05
15×15	19	2.46	11	1.99	8	1.94	6	1.85	6	2.20	5	2.17
31×31	41	3.95	24	3.46	18	3.44	14	3.35	12	3.45	11	3.68
63×63	81	6.02	51	5.15	36	4.77	30	4.92	26	5.09	23	5.24
127×127	213	7.05	130	5.03	104	5.08	84	5.01	79	5.53	68	5.46

回数は3回から4回が計算時間が最も短くなっていることがわかる。スムージングの反復回数を1回づつ増やしていくたびに反復回数が減少し、その減少のしかたはスムージングの反復回数を増やすたびに減少していることがわかる。また1×1まで粗くする場合を除いてはスムージングの反復回数を1回から2回に増やしたときには反復回数は60%程に減少している。

1×1まで粗くした場合だけはスムージングの反復回数を増やすたびに計算時間も増加していることがわかる。これは前述のように粗い格子点になればなるほどidleのプロセッサーが増加し、通信のオーバーヘッドが増加してくるために起こる現象である。

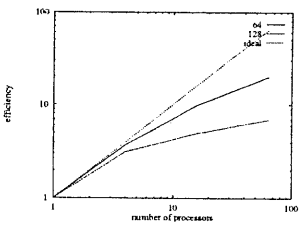
この場合にはプロセッサー台数よりも少ない数の格子点まで粗くする場合には通信のオーバーヘッドを考慮に入れてスムージングの反復回数は1回にするのが良く、粗い格子点をあまり用いない場合には収束性からスムージングの反復回数は2,3回が適当であると考えられる。

3.3 台数効果

この章では扱うデータの個数を固定し、使用するプロセッサー台数を変化させたときの計算時間を測定している。MG前処理にはVサイクルでスムージングにRBGS法2反復を用いたものを使い、プロセッサー当たり1個のデータになるまで粗くしている。実験の結果は表3に示されている。非対称の係数行列の場合、対称

表3. 台数効果

問題の大きさ	反復回数	1×1(台)	2×2	4×4	8×8
63×63	7	6.48	2.07	1.30	0.941
127×127	7	29.5	7.92	3.01	1.50
255×255	6		29.0	7.92	2.69



行列のときに観察されたような([2],[5])スーパーリニアの効果は得られていない。

3.4 MGCGS 法と MILUCGS 法の比較

MILUCGS 法ではデータの分割は MG 法と違って境界上にあるデータをそれに隣接するプロセッサーが重複して持つ必要がないので、前の実験とはデータの分割の仕方が違うことに注意してもらいたい。それにともなって問題の大きさが異なっていることに注意してもらいたい。使用した問題は MGCGS 法のときと同じ問題を使っている。またプロセッサー台数は 64 台である。その結果を表 4に示す。表 4と表 3の 8×8 の欄

表 4. MILUCGS 法

問題の大きさ	反復回数	計算時間 (秒)
64×64	36	1.91
128×128	70	7.09
256×256	137	42.9

を見比べることによって、従来使われてきた MILUCGS 法よりもはるかに短い計算時間で MGCGS 法は収束していることがわかる。しかもそれは大規模な問題になればなるほど顕著になっていることがわかる。この原因は MILUCGS 法の場合には問題が大きくなるにつれ収束までの反復回数が急激に増加することにある。MGCGS 法の場合にはこのような反復回数の増加は見られない。なお MILU 前処理は格子点を Red-Black に色わけして並列性を高めることができるが、そのようなことは収束性が悪化するために行なっていない。

4 まとめ

この研究では非対称行列に対して MG 前処理を行ない、アルゴリズムを並列計算機上に実装することによって効率の良い MG 前処理を考察した。MG 法では粗い格子点上における係数行列の非対称性の増加を防ぐために風上差分法を使用すると、中心差分法の場合に比べて収束性の悪化が見られるものの、MGCGS 法では収束性の悪化は見られない。そのためにそのような離散化を行ない高並列なスムージングを行なっても収束性は悪くならない。さらに使用する並列計算機の能力を十分に発揮できるところまで粗いグリッドを使えば MGCGS 法は高並列で非常に収束性の良い方法であることを示した。さらに従来用いられてきた MILUCGS 法よりも効率の良い解法であることを示した。

参考文献

[1] Wolfgang Hackbusch. *Multi-grid Methods and Application.* Springer Verlag, 1985.

[2] Osamu Tatebe. "The Multigrid Preconditioned Conjugate Gradient Method". In *the proceedings of Sixth Copper Montain Conference, NASA CP*, 1993 (to appear).

[3] Pieter Wesseling. *AN INTRODUCTION TO MULTIGRID METHODS.* JOHN WILEY AND SONS, 1991.

[4] Jianping Zhu. *SOLVING PARTIAL DIFFERENTIAL EQUATIONS ON PARALLEL COMPUTERS.* World Scientific, 1994.

[5] 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の並列化". JSPP '93 論文集 pp.387–394, May 18 1993.

[6] 襲田 勉 建部 修見 小柳義夫. "マルチグリッド前処理付き自乗共役勾配法". 情報処理学会研究報告 93-HPC-48 pp.65–71, Aug 19,20 1993.

[7] 村田 健郎 名取 亮 唐木幸比古. 大型数値シミュレーション. 岩波書店, 1990.