Vol.2011-HPC-129 No.21
2011/3/16

GPGPU を用いた AMG 法

高橋光佑† 藤井昭宏† 小柳義夫†

GPU 上での AMG 法の計算性能の向上を目指し，複数種類の緩和法を合わせてスムーザとして適用し評価した．考慮した緩和法はヤコビ法，マルチカラー・ガウス・ザイデル法，並列性を高めるために依存関係の一部を無視したマルチカラー・ガウス・ザイデル法である．各手法について GPU のコアレッシングの条件を満たすような実装を提案した．GPU 上に於いて三次元拡散方程式の等方性問題では，ヤコビ法でも性能は出るが，異方性問題の場合ではガウス・ザイデル法を適用した方が良い結果となった．また 8 コア環境の CPU 版との性能比では，どちらの場合も GPU 版が 4 倍程度の高速化を示し，本手法の有効性を確認した．

GPGPU-based AMG Method

Kosuke Takahashi†, Akihiro Fujii† and Yoshio Oyanagi†

Various types of smoothers are evaluated for GPU-based AMG solvers. We implemented Jacobi method, multi-color Gauss-Seidel method and modified multi-color Gauss-Seidel method which colors unknowns with smaller number of colors by ruling out weak dependencies. This paper shows these three smoothers' implementation which takes advantage of GPU coalescing memory access. Although Jacobi smoother works well for isotropic Poisson problems, some types of Gauss-Seidel smoother make GPU-based AMG solver faster for anisotropic problems in our numerical test. GPU-based AMG solver becomes four times faster than AMG solver on quad-core 2CPU.

1. はじめに

定常物理拡散など様々な物理現象は大規模な連立一次方程式を解くことに帰着される. AMG 法はこれらの連立一次方程式を高速に解く手法の一つとして知られており，構築部と解法部から成る．

GPGPU とは GPU を用いた汎目的計算技術で，近年，高速化手法として注目されている．従来，性能を引き出すために高い並列性が求められる GPGPU に合わせ，緩和法には並列性の高い JC 法が用いられてきた[1]．しかし JC 法は GS 法と比べ収束が悪く，連立一次方程式の性質によっては収束が困難な場合が存在する．

本研究ではガウス・ザイデル法（以下，GS 法）をマルチカラーにより GPU 上で並列化し，これを緩和法に用いた代数的多重格子法（以下，AMG 法）の解法部全体を GPU に実装した．緩和法にヤコビ法（以下，JS 法）を用いた場合と比較を行った．同時に 8 コア環境の対 CPU 比で等方性異方性問わず 4 倍程度の高速化を実現した．

2. マルチカラー・ガウス・ザイデル法

並列化処理に於いて，データのアクセス競合が起こる場合が存在する．これは計算を実行する度に異なる挙動を招く原因と成る（Chaotic 性）．JC 法では各並列処理間にデータのアクセス競合が起こる可能性がないため，高い並列性が存在する．しかし GS 法では連立一次方程式の解に対してアクセス競合が起こる可能性が存在し，そのまま並列処理を行うと Chaotic-GS 法になってしまう．

ここで，疎行列問題の場合には必ずしも全ての未知数に依存関係が存在するわけではない事に注目し，依存関係が存在しないようにグループ分けする．これはグラフ彩色問題に帰着することが出来るため，Multi Color 法と呼ばれる[2]．本研究では彩色方法に貪欲彩色法と Welsh-Powell 法を採用した．

Welsh-Powell 法は貪欲彩色法の一種で，最大次数の一つ上の数で彩色可能なことが保証されている．色数は連立一次方程式の構造によっては肥大化し，十分な並列性が得られない場合が考えられる．図 2-1 は元の疎行列の非ゼロ要素の分布図，図 2-2 は完全な彩色後に各行各列を並べ替えた後の様子を示している．彩色後の対角にある四角形の数が色数となる．色数が規模に対して多い様子が確認できる．色数を少なくする工夫として，依存関係の大小に応じて彩色の際の考慮対象から除外する手法が考えられる．各行成分の絶対値のうち最大のものを探索し，その値の 9 割を超える非ゼロ要素のみ彩色の際に考慮した場合を図 2-3 に示す．この場合は 2 色で塗り分けられている．これにより並列性は高まるが，Chaotic 性が発生する．

† 工学院大学
Kogakuin University

Vol.2011-HPC-129 No.21
2011/3/16

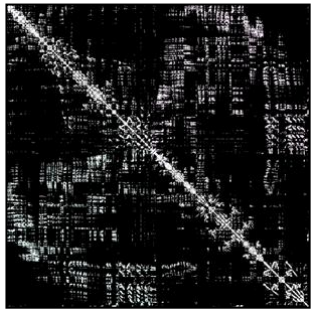

図 2-1　疎行列の非ゼロ要素分布

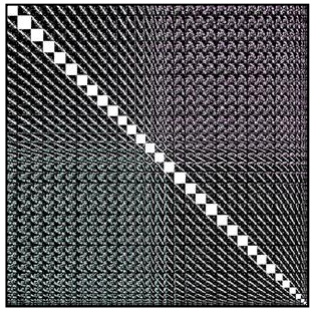

図 2-2　完全な彩色後，色ごとに並べ替えた後の様子

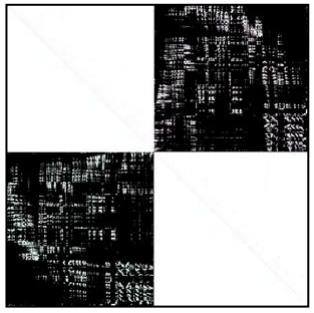

図 2-3　調整した不完全な彩色後，色ごとに並べ替えた後の様子

3. GPGPU を用いた AMG 法

3.1 AMG 法

解法部は主に内積計算，行列ベクトル積，緩和法から成り立っている．内積計算は残差誤差のノルム計算のために利用され，行列ベクトル積と緩和法は V-cycle 内の計算で利用される．V-cycle の内容例は図 3-1 のようになっている．AMG 法は階層型で，最上層に元の問題が設置され，階層が下がるに連れて元より粗く小さい問題が設置されている（階層数は可変）．階層移動には構築部で用意した Prolongation 行列と Restriction 行列を使う．階層を降りる際には現階層での残差誤差を計算し，横長の Restriction 行列と行列ベクトル積を行うことで短いベクトルを生成し，一つ下の階層で利用する．階層を上る際は現階層の解と縦長の Prolongation 行列の行列ベクトル積を行うことで長いベクトルを生成し，一つ上の階層の補正解として利用する．

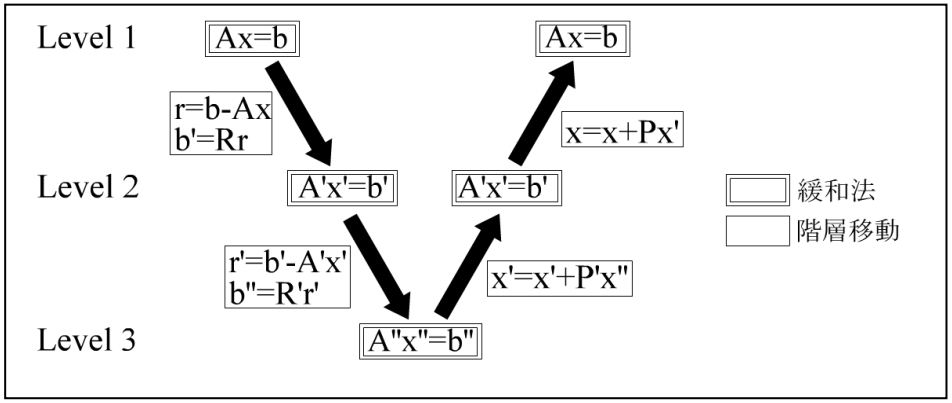

図 3-1　V-cycle の概要図

ここで本研究での GPU 上での行列ベクトル積実装について触れる．GPU の性能を最大限引き出す場合，GPU が持つ「グローバル・メモリ」へのアクセスに注意を払う必要がある．GPU にはデータ転送を効率的に行う仕組み「コアレッシング」が存在する．詳細は後述するが，今回用いる GPU は Fermi アーキテクチャの Tesla C2050 であり，「コアレッシング」が利用される条件として「16 個のスレッドがアクセスするアドレスが連続した領域にまとまっている」，「先頭スレッドのアクセスするアドレスが 128 バイト境界にアラインメントされている」という二点がある．故にパディングなどを用いてデータのアラインメントを図る．通常の行列ベクトル積の場合はアラインメントの調整だけで「コアレッシング」の条件を満たすが，マルチカラー・ガウス・

ザイデルを用いる場合は更に対策が必要となる．色ごとに処理を行う際，そのまま行列やベクトルを用いると各スレッドがアクセスするアドレスが連続した領域に存在せず,「コアレッシング」の条件を満たすにはデータを再編する必要がある．圧縮された疎行列のデータの具体的な再編は次節で説明する．

GPU 上での内積計算の実装について触れる．本研究では GPU 上の「シェアード・メモリ」を用いて高速化した内積計算を利用している．「シェアード・メモリ」を用いることで総和計算を高速化することが出来る．「シェアード・メモリ」が「ブロック内のスレッドから共有」であることを利用し,「シェアード・メモリ」のリダクションを行うことで高速化する．詳細な実装については文献 3 を利用した．AMG 法の収束判定を行う際，残差からノルムの計算を行うため，総和計算の高速化は重要である．また後述するが本研究では AMG 法を前処理に用いた Bi-CGSTAB 法についても評価を行うが，Bi-CGSTAB 法では内積計算が頻繁に行われる．

本研究では最下層部分のみ CPU 上で実装した．GPU は大規模な問題に対しては計算性能に期待が持てるが，小規模な問題に大しては期待が持てない．AMG 法では粗いレベルの問題が小規模な問題なため, 粗いレベルも GPU 上で実装するとボトルネックになりかねない． GPU 側で最下層まで Restriction された値を CPU 側に読み込み，直接法を用いて計算した後，GPU 側に転送し Prolongation する．GPU 上で小規模な問題を直接法で高速に解くことは困難なため, 全てを GPU で実装するより全体として計算性能は向上する．

最上層以外では疎行列の非ゼロ要素の構成が複雑となり，彩色した際の色数が多くなる．これは並列性の低下を招き，計算速度が降下する．前述したように，強制的に色数を減らすことにより計算速度は確保されるが，Chaotic 性が生じてしまう．ひとつの解決法として，各階層に応じで緩和法を使い分けることにした．例えば上から二つの階層までは GS 法で実装し，残りの階層は JC 法で実装するといった具合だ．

3.2 マルチカラー・ガウス・ザイデル法

彩色後，疎行列の要素を並べ替えて GPU に転送する必要がある．ここで，コアレッシングを考慮した並べ替えとパディングを行う．実行時には色ごとに並列処理が実行される．つまりコアレッシング条件を満たすためには色ごとに疎行列がまとめられている必要がある. 疎行列の圧縮形式として主に CRS 形式と CCS 形式が挙げられるが，ここでは CRS 形式を基準に話を進める．

例として図 3-2 のような疎行列を考える．各スレッドには各行成分に関する計算を担当させる. 同じ色に彩色された行に関しては同時に処理が行え, 並列性が存在する．例の場合，黄色に彩色された行要素について最大 5 並列まで実行が可能だ．図 3-3 は CRS 形式で疎行列を圧縮した例である．ここで id とは，色ごとに並列処理を実行した場合のスレッド ID を想定している．そのままの CRS 形式の場合，「16 個のスレッドがアクセスするアドレスが連続した領域にまとまっている」点を満たしていないので，コアレッシングは利用されない．図 3-4 はパディングを用いて仮に 4 要素にアラインメントし，スレッドがアクセスするアドレスが連続した領域になるように並べ替えた例である．

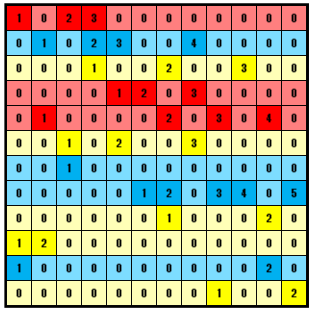

図 3-2 疎行列彩色後例

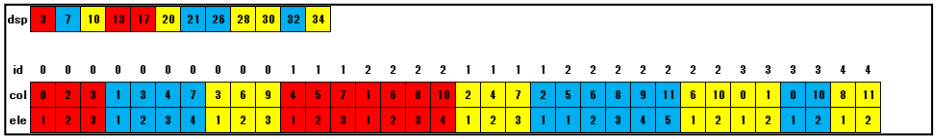

図 3-3 疎行列 CRS 形式

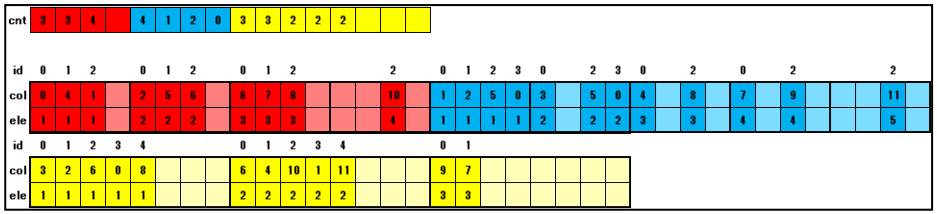

図 3-4 疎行列特製圧縮形式

ここで並べ換えとパディングを行ったことにより，列番号を変更する必要が出てくる．今回の例の青色についての書き換えを図 3-5 に示す．図中の XB とは連立一次方程式の x ベクトルと b ベクトルのことである（並べ替えが同じであるので一括記載）．緩和法や行列ベクトル積を計算する際，疎行列の要素の列番号に応じて x ベクトルや

b ベクトルの要素が呼び出されるので, 新しい並びに対応した col に書き換える必要がある.

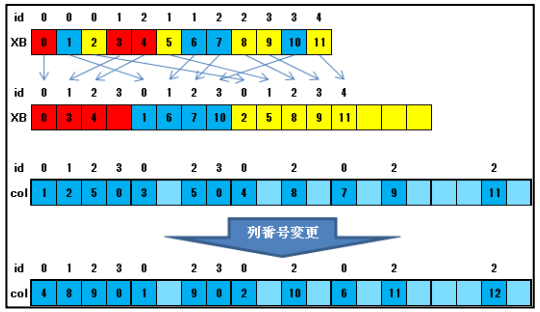

図 3-5　列番号の書き換え

本節と関連して，AMG 法の Prolongation 行列と Restriction 行列に関しても同様なデータの並べ換えと書き換えが必要になる．ここで注意しなければならないのは，連立一次方程式の疎行列 A に関しては同階層の彩色のみ考慮するのに対し，階層間に存在する疎行列の Prolongation 行列と Restriction 行列は，行と列に関して別階層の彩色を考慮してデータを整形する．

GPU 上で JC 法や GS 法を実装する場合，計算時間の殆どが「グローバル・メモリ」へのアクセスが占める．四則演算などを行う際と比べて「グローバル・メモリ」へのアクセスのレイテンシが 100 倍以上あるためである．疎行列の場合，「グローバル・メモリ」から読み込んだ値の再利用性も低いため，レイテンシの隠蔽も困難である．故に計算性能は演算速度ではなくメモリ帯域幅で決まるが，無駄のないアクセスを行うためにもコアレッシング対策を行うことが計算性能の向上に直結する．

4. 数値実験と考察

4.1 実行環境と計算対象

本研究の実行環境を表 4-1 に示す．比較に使用する CPU は Quad-core が 2 つなので最大 8 並列まで実行可能である．GPU の詳細を表 4-2 に示す．GPU の理論値性能は倍精度演算で 515Gflosp，メモリ帯域幅が 144GB/s である．今回の手法はメモリ帯域幅を使い切ることで計算性能が向上するので，後者のほうが重要である．

表 4-1　実行環境

CPU	Intel Xeon X5570 Quad-core 2CPU 2.93GHz
Graphics	ETS2050-C3ER
Memory	12GB (6x 2GB DDR3-1333 DIMM)
OS	CentOS 5.5 (64bit)

表 4-2　GPU の仕様

GPU Chip	NVIDIA TESLA C2050
Peak Performance	515Gflops
Number of CUDA core	448
CUDA core Clock	1.15GHz
Memory Transfer	144GB/s
Memory Interface	384bit
Video memory	3GB GDDR5

問題は三次元拡散方程式で等方性と異方性のものを用意．異方性は Z 軸方向の拡散が他軸より 0.01%の拡散になっている．問題サイズは等方性が10^3～70^3，異方性が10^3～60^3のものを用意．いずれも貪欲彩色法で 8 色に塗り分け可能である．

4.2 比較手法

比較は対称 GS 法を用いた CPU 版 GS based-AMG 法, JC 法を用いた GPU 版 JC-based AMG 法，最上層のみ GS 法を用いた GPU 版 GS-JC-mixed AMG 法，Chaotic 性のある GPU 版 Chaotic GS based-AMG 法で比較を行った．なお，AMG 最下層は CPU 版が対称 GS 法を 30 回，GPU 版が LU 分解の前進代入と後退代入で実装している．CPU 版は分割領域に基づいて MPI で実装している[4]. 並列度を変えて性能が最大になるように実行した．最大 8 並列で実行している．パラメータは総当り的に複数回実行して最良のものを採用(表 4-3). また, 以上の手法の AMG 法を前処理として使う Bi-CGSTAB 法（以下，AMGBi-CGSTAB 法）についても GPU 版のみ比較を行った．こちらもパラメータは総当り的に複数回実行して最良のものを採用．採用例を表 4-4 に示す．後述するが AMGBi-CGSTAB 法の異方性問題は収束しなかったため，パラメータ無しで記載されている．Iso-70^3 は70^3のサイズの等方性問題，Ani-60^3 は60^3のサイズの異方性問題を表す．「Lv.」は各階層を示し，「Lv.0」が最も細かく，「Lv.4」が最も粗い階層である．収束判定は 2-ノルム相対誤差が10^{-7}未満になった時とした．

表 4-3　AMG 法の各パラメータ

Iso-70^3	緩和法反復回数					緩和係数				
	Lv.0	Lv.1	Lv.2	Lv.3	Lv.4	Lv.0	Lv.1	Lv.2	Lv.3	Lv.4
GS-JC-mixed	1	1	2	6		0.65	0.45	1.00	1.00	
JC-based	1	1	2	6		0.90	0.45	1.00	1.00	
Chaotic-GS-based	1	1	2	6		0.85	0.45	1.00	1.00	
CPU-GS-based	1	1	1	1	30	1.00	1.00	1.00	1.00	1.00
Ani-60^3	緩和法反復回数					緩和係数				
	Lv.0	Lv.1	Lv.2	Lv.3	Lv.4	Lv.0	Lv.1	Lv.2	Lv.3	Lv.4
GS-JC-mixed	5	1	2	4		1.50	0.50	0.50	0.50	
JC-based	1	1	1	1		0.50	0.50	0.50	0.50	
Chaotic-GS-based	5	4	2	1		1.30	1.00	1.00	1.00	
CPU-GS-based	2	2	2	2	30	0.75	0.75	0.75	0.75	0.75

表 4-4　AMGBi-CGSTAB 法の各パラメータ

Iso-70^3	緩和法反復回数					緩和係数				
	Lv.0	Lv.1	Lv.2	Lv.3	Lv.4	Lv.0	Lv.1	Lv.2	Lv.3	Lv.4
GS-JC-mixed	1	1	2	6		1.00	1.00	1.00	1.00	
JC-based	1	1	1	1		1.00	1.00	1.00	1.00	
Chaotic-GS-based	1	1	2	6		1.00	1.00	1.00	1.00	
Ani-60^3	緩和法反復回数					緩和係数				
	Lv.0	Lv.1	Lv.2	Lv.3	Lv.4	Lv.0	Lv.1	Lv.2	Lv.3	Lv.4
GS-JC-mixed	8	1	1	1		1.00	1.00	1.00	1.00	
JC-based										
Chaotic-GS-based	8	4	0	0		1.00	1.00	1.00	1.00	

4.3 計算結果

まず AMG 法単体の実行結果を図 4-1，図 4-2 に示す．等方性問題の場合，JC 法を用いた AMG 法が総合的に良い結果となった．各階層の緩和部の反復回数は，どの手法についても低レベルほど回数を増やしたほうが計算性能は向上．異方性問題の場合，Chaotic-GS 法を用いた AMG 法が総合的に良い結果となった．JC 法を用いた場合が大きく性能を落としている．Chaotic-GS 法を適用した場合の反復回数の変動を確認するため，1000 回のシミュレートを行ったが，最終残差の変動以外は確認できなかった．

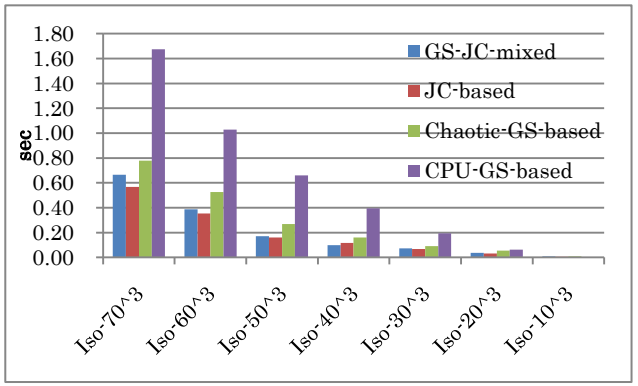

図 4-1　等方性問題 AMG 法 反復部 計算時間

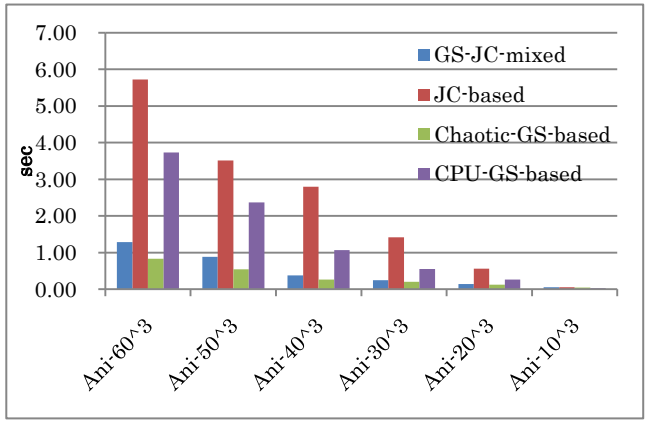

図 4-2　異方性問題 AMG 法 反復部 計算時間

続いて AMGBi-CGSTAB 法と単体の AMG 法の実行結果の比較を表 4-5 に示す．本研究で扱った最大問題サイズのみの比較を示した．等方性の問題ではどの緩和法を用いても AMGBi-CGSTAB 法が良い結果となったが，異方性の問題では GS 法と JC 法の両方を緩和法に用いた場合，同等か，AMG 法が良い結果となった．特に JC 法を用いた場合は発散してしまい，収束することは無かった．単体の AMG 法と同様，Chaotic-GS 法を適用した場合の反復回数の変動を確認するため，1000 回のシミュレートを行ったが，最終残差の変動以外は確認できなかった．

表 4-5　AMGBi-CGSTAB 法と AMG 法 反復部 計算時間

Iso-70^3	GS-JC-mixed	JC-based	Chaotic-GS-based
AMGBiCGSTAB	0.46sec	0.43sec	0.62sec
AMG	0.63sec	0.57sec	0.78sec
Ani-60^3	GS-JC-mixed	JC-based	Chaotic-GS-based
AMGBI-CGSTAB	1.26sec	収束せず	1.17sec
AMG	1.28sec	5.72sec	0.83sec

4.4 考察

単体の AMG 法について考察する．等方性問題の場合，緩和法の手法による収束の変化が少ないことが表 4-6 から分かる．GS 法は JC 法に比べ，並列性が低くなるために一回反復毎の計算時間が増えてしまう．その代わりに収束が安定することが特徴としてあるが，収束の安定による計算量の減少より一回反復毎の計算時間の増加がネックになってしまう．異方性の場合は逆に緩和法の手法よる収束の変化が大きく，全体の反復回数に大きく差が出ているのが表 4-6 から分かる．この異方性は二次元の問題が連層になっているような構造をしているので，GS 法による収束の安定が非常に大きい．

表 4-6　AMG 法 反復部 反復回数

Iso	GS-JC mixed	JC based	Chaotic-GS-based	CPU-based	Ani	GS-JC mixed	JC based	Chaotic-GS-based	CPU-based
70^3	70	66	56	42					
60^3	57	59	52	38	60^3	89	1159	39	98
50^3	40	43	38	38	50^3	98	1252	40	97
40^3	36	47	45	40	40^3	79	1644	41	91
30^3	36	41	36	36	30^3	73	1339	40	85
20^3	36	10	35	26	20^3	47	968	25	70
10^3	12	32	12	13	10^3	33	250	27	46

AMGBi-CGSTAB 法について考察する．本研究では Bi-CGSTAB 法の前処理として一回の反復につき二回の AMG 法が実行されている．さらに Bi-CGSTAB では一回の反復につき二回の行列ベクトル積の計算も行う．つまり AMGBi-CGSTAB 法の導入により AMG 法と比較して反復回数が大きく減少しなければ計算時間は減らない．反復回数の比較を表 4-7 に示す．等方性問題は緩和法の反復部回数が近いため，30%に減少する程度で良い性能が出た．異方性問題の場合，AMGBi-CGSTAB 法は収束を良くするために AMG 法より多く細かいレベルに対して緩和法を施行するため，それが不利に働いて計算性能が伸び悩んだ．

表 4-7　AMGBi-CGSTAB 法と AMG 法 反復部 反復回数比較

Iso-70^3	GS-JC-mixed	JC-based	Chaotic-GS-based
AMGBi-CGSTAB	17(24%)	21(31%)	18(32%)
AMG	70	66	56
Ani-60^3	GS-JC-mixed	JC-based	Chaotic-GS-based
AMGBi-CGSTAB	26(29%)	収束せず	18(46%)
AMG	89	1159	39

5. おわりに

本稿では AMG 法の GPU 上での効率的な実装手法について考察した．特にマルチカラー・ガウス・ザイデル法の GPU 上での効率的な実装手法を提案し，AMG 法に組み込んで評価を行った．

数値実験では AMG 法の各レベルに於いて従来のようにヤコビ法を適用した場合，最も細かいレベルのみマルチカラー・ガウス・ザイデル法を適用した場合，更に接続の弱い部分を無視して全てのレベルで Chaotic ガウス・ザイデル法を適用した場合を GPU 上で実装し，CPU 版と比較を行った．

その結果，8 コア環境での CPU 版 AMG 法と比較した場合，GPU 版 AMG 法は等方性の問題で 4 倍，異方性の問題で 4.5 倍の計算性能が出た．また等方性の問題ではヤコビ法で問題ないが，強い異方性の入った問題では GPU 上でもマルチカラー・ガウス・ザイデル法を適用すると，より計算性能が出ることが分かった．

Bi-CGSTAB 法の前処理としても評価をしたところ等方性の問題では単体の AMG 法より計算性能が出たが，異方性の問題では単体の AMG 法が良い場合を確認できた．また AMGBi-CGSTAB 法でも強い異方性の入った問題ではマルチカラー・ガウス・ザイデル法を適用すると，より計算性能が出ることが分かった．

Vol.2011-HPC-129 No.21
2011/3/16

今後はマルチ GPU 環境での性能評価や, 様々な疎行列問題に対しての性能評価を行って行く必要がある.

参考文献

1) G. Haase, M. Liebmann, C. Douglas, and G. Plank. :A Parallel Algebraic Multigrid Solver on Graphical Processing Units, HPCA-16, LNCS 5938, pp.38-47, Bangalore, India, January (2010)
2) 佐藤陽平: マルチカラー･ガウス･ザイデル法を用いた非構造格子粘性流体解析コード SURF の並列化，第 8 回海上技術安全研究所研究発表会，8th，pp.319-320，June (2008)
3) 青木尊之・額田彰: はじめての CUDA プログラミング，工学者，(2008)
4) 藤井昭宏，小柳義夫: 科学技術シミュレーションにて多用される代数的多重格子法の評価，シミュレーション 第 28 巻第 4 号，pp.9-14，(2009)