抽出したニアカーネルベクトルを複数本設定するSA-AMG法の性能評価

野村直也

東京大学

1．はじめに

科学技術計算の分野では，大規模連立一次方程式$\boldsymbol{Ax} = \boldsymbol{b}$は様々な場面で現れ，高速に解くことを求められることが多い．そこで，大規模連立一次方程式を高速に解くことができる手法として，AMG(Algebraic Multigrid)法が提案されている[1]．AMG法は問題行列から階層的に，次元数の異なる行列を生成して解く手法である．AMG 法の中の解法のひとつに，Smoothed Aggregationに基づくAMG法がある[2][3][4]．この手法は，多くの問題に対して有用であることが知られており，広く用いられている解法となっている．本研究では，このSA-AMG法を対象とし研究を行っている．

SA-AMG法は大きく分けて，問題生成部（以下，構築部）と反復解法部（以下，解法部）に分かれている．構築部では，問題行列に基づくグラフ構造を作成し，それを基にアグリゲートと呼ばれる節点集合を作成する．そして，作成したアグリゲートを基に粗い問題を作成する．これを再帰的に行うことで，次元数のより小さい複数の行列を作成する．解法部では，構築部で階層的に生成された行列に対し，Gauss-Seidel 法などの緩和法を用いて問題行列を解く．このとき，元の問題行列などの大規模な行列が設置される階層（レベル）を細かいレベル，問題行列から生成された小規模な行列が設置される階層（レベル）を粗いレベルと呼ぶ．

SA-AMG 法は収束しにくい成分を粗いレベルで解くことにより，高い収束性を実現している．ここで，収束しにくい成分とは，一般にニアカーネルベクトルと呼ばれ，$\boldsymbol{Av} \approx \boldsymbol{0}$になるようなベクトル$\boldsymbol{v}$のことをさす．一般的な定常反復解法（例：Gauss-Seidel 法）で解いた際，この成分が解$\boldsymbol{x}$に存在することにより，残差が停滞し，効率よく解を求められない原因となる．SA-AMG法では，構築部においてニアカーネルベクトルを用いて粗い問題を生成することにより，小さい問題行列で効率よく収束しにくい成分を解くことができ，残差を効率よく収束させることが可能となる．

本研究では，3 次元弾性体の問題を使用しており，この問題では平行移動成分と回転成分がニアカーネルベクトルとして知られている．SA-AMG法においてこれらのニアカーネルベクトルを設定することにより，反復回数と実行時間の双方で改善がみられることが知られている．そこで本研究では，問題行列に応じた適切なニアカーネルベクトルの設定手法と，その効果についての検証を行う．適切なニアカーネルベクトルを設定するための手法を見つける最初の段階として，本研究ではαSA法[5]をもとに，V-cycleを用いて問題行列からニアカーネルベクトルを複数本抽出する手法を実装した．そして，その手法で抽出したニアカーネルベクトルの設定本数を変化させることで，平行移動成分と回転成分を設定する手法と比べ，性能改善がみられるかを検証した．

２．SA-AMG 法

SA-AMG 法は AMG 法の中の解法のひとつであり，与えられた問題行列から階層的に小規模な問題行列を生成し，それらを用いて解く手法である．この章ではまず，AMG 法についての説明を行い，次に SA-AMG 法についての説明を行う．

AMG 法は，大規模な非構造格子による線形問題を高速に解く数値解法のひとつである．AMG 法ではまず，与えられた問題行列を複数段階に分けて小規模な行列を生成し，これらを用いて問題行列を解く．AMG 法は大きく分けて構築部と解法部の 2 つの処理からなる．構築部は問題行列から未知数間のグラフ構造を作り，次のレベルに残す未知数を選択する．そして，粗いレベルの行列を生成する．これを再帰的に行うことで，階層的に生成された行列を作る．一方，解法部は構築部で生成された行列を用い，実際に問題を解く．

構築部の構造を図 1 に示す．構築部では細かいレベルの問題行列を基に，粗いレベルの問題行列や補間演算子である Prolongation (P) 行列と Restriction (R) 行列を階層的に生成する．AMG 法は階層型で，最上階に与えられた問題行列を用い，階層が下がるにつれて問題行列より小規模な行列を生成する.階層数は問題サイズによって可変となる.粗いレベルの問題行列は，横長の Restriction 行列と縦長の Prolongation 行列,さらに現階層の問題行列とで行列行列積（RAP）を行うことで作成している．

解法部の構造を図 2 に示す．解法部は，主に行列ベクトル積と緩和法から成り立っている．この図では，最上層に与えられた問題行列が設置され，階層が下がるにつれて，問題行列より小規模な行列が設置される．階層数は問題サイズによって可変となる．階層移動では，構築部で用意した補間演算子の Prolongation 行列と Restriction 行列を使う．階層を下りる際には，現階層の残差を計算し，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，ひと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 行列との行列ベクトル積を行うことで長いベクトルを生成し，ひとつ上の階層の補正解として利用する.複数の階層を行き来する様子が V 字を連想させるため,このような解法部を V-cycle と呼ぶ.

補間演算子から行列の階層構造が生成されるため，この補間演算子の生成手法により様々な AMG 法が存在する[6]．本研究では，その中でも SA-AMG 法を対象とする．この手法は問題行列のみから未知数間の依存関係を定義する．そして，依存関係のある未知数同士で集合を作り，その集合内で重みづけをして補間演算子を生成する.SA-AMG 法はさまざまな分野で利用されており，AMG 法の代表的な手法のひとつとなっている．

SA-AMG 法では,問題行列に基づく節点と辺で構成されたグラフ構造を用いて粗い問題を作成していく．ここで，問題行列の各行が節点に対応し，非ゼロ要素が辺に対応している．粗い問題を作成する際に，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．アグリゲートは図 3 のように，次の粗いレベルで 1 つの節点に対応し，グラフ構造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．任意の節点がどこか 1 つのアグリゲートに属するように，アグリゲートが生成される．このアグリゲート内の未知数に重み付けをすることで補間演算子を計算し,行列の階層構造を作成する.補間演算子である Prolongation 行列と Restriction 行列は，行列の階層構造を作成する際に用いられる．これらの行列の生成にニアカーネルベクトルを用いることで，収束性をさらに高めることができる．このことについては，次節で説明を行う．

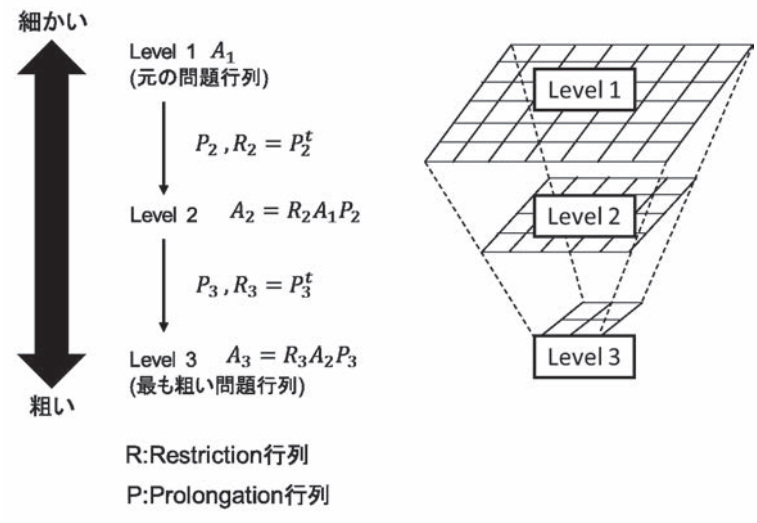

図 1　構築部

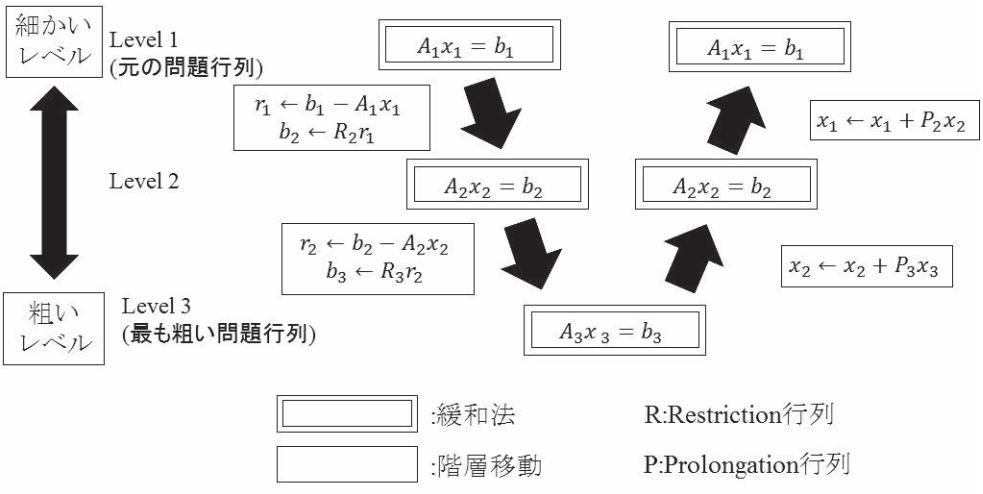

図 2　解法部

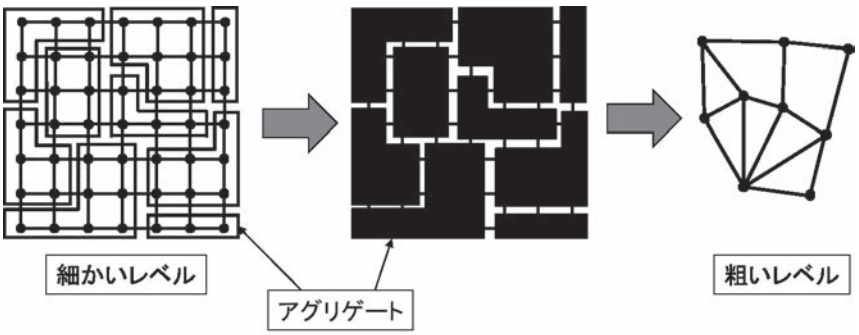

図 3　アグリゲート生成

３．ニアカーネルベクトル

この章ではニアカーネルベクトルの定義と，SA-AMG 法においてニアカーネルベクトルがどのように用いられるかを説明する．

ニアカーネルベクトルとは，問題行列 A との行列ベクトル積が 0 に近くなる 0 ベクトルでないベクトル（ニアカーネルベクトルを $\boldsymbol{v}$ とおくと，$A\boldsymbol{v}\approx 0$，$\boldsymbol{v}\neq 0$）のことを言う．一般的な定常反復解法（例：Gauss-Seidel 法）を用いる場合，残差（$\boldsymbol{b}-A\boldsymbol{x}$）を基に解の修正を行うが，ニアカーネルベクトルの影響により $A\boldsymbol{x}$ が 0 に近くなり，残差が停滞してしまう可能性がある．

SA-AMG 法では，補間演算子である Prolongation 行列と Restriction 行列にニアカーネルベクトルを用いることで，粗いレベルに収束しにくい成分を移動させることができる．これにより，粗いレベルで収束しにくい成分を解くことができるため，効率よく解を収束させることが可能となる[2][3][4]．

問題の性質からニアカーネルベクトルをある程度特定できる場合もある．例えば，本研究で用いている弾性体の問題では，平行移動成分と回転成分がニアカーネルベクトルとして知られている．弾性体の問題を対象としている場合，これらを SA-AMG 法に用いることで，収束性が高まることが知られている．

ニアカーネルベクトルの補間演算子への設定方法を図 4 に示す．図 4 は 2 本のニアカーネルベクトルを用いて，問題$\boldsymbol{A}$から 2 つのアグリゲートを作成するときの，Prolongation 行列の作成方法を図示している．この図のように行列$\boldsymbol{P}$を生成する際には，アグリゲートの節点番号に対応した場所のニアカーネルベクトルの要素を抜き出し，それを並べて QR 分解し，スムーザをかけることで Prolongation 行列を作成する．SA-AMG 法では，ニアカーネルベクトルを複数本設定することができる．その場合，抜き出す際のベクトル本数が増え補間演算子の行列が大きくなり，結果として計算量が増えることとなる．そのため，ニアカーネルベクトルの本数と実行時間でトレードオフが発生すると考えられる．

図 5 と図 6 に，SA-AMG 法においてニアカーネルベクトルを複数本設定することによる反復回数の変化を示す．図 5 にこの実験で対象とした問題を示す．この問題は 3 次元弾性体問題である．弾性体の問題については，5 節で詳しく説明する．この弾性体の問題は，上半分が柔らかい物体に対して，ある一部分に力を加え，どのように変形するかを解く問題となっている．また，ヤング率を上半分が 0.8，下半分を 1 に設定している．反復の終了条件は相対残差が 1.0×10^{-7} となったときとした．また，問題サイズについては，1 プロセスあたり 6×15×60 としたウィークスケーリングで計測を行った．図 6 に，収束までに必要とした反復回数のグラフを示す．グラフの横軸はプロセス数であり，縦軸は反復回数となっており，下に行くほど反復回数が少なく，収束性が良いことを示している．また，グラフの要素である“Number of kernel vectors: 1” はすべての要素が 1 の定数ベクトルをニアカーネルベクトルとして設定した場合，“Number of kernel vectors: 3” は平行移動成分のみを設定した場合，“Number of kernel vectors: 6” は平行移動成分と回転成分を設定したときのグラフとなっている．図 6 より，ニアカーネルベクトルを複数本設定することで，収束性の改善がみられることがわかる．これは，適切なニアカーネルベクトルが設定できているため，このような傾向がみられたと考えられる．本論文では，ニアカーネルベクトルを問題行列から抽出することにより，これらのベクトルよりもさらに反復回数の改善が可能となる性質のよいニアカーネルベクトルが設定できるかの検証も行う．

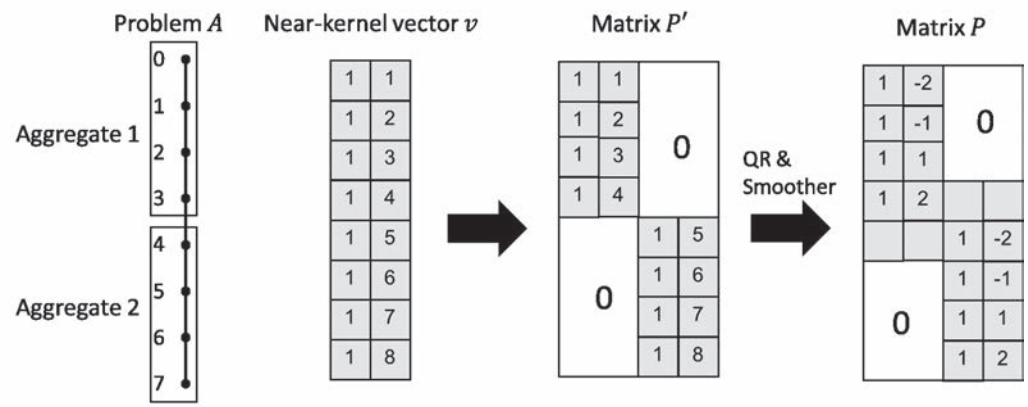

図 4　ニアカーネルベクトルを用いた Prolongation 行列の作成方法

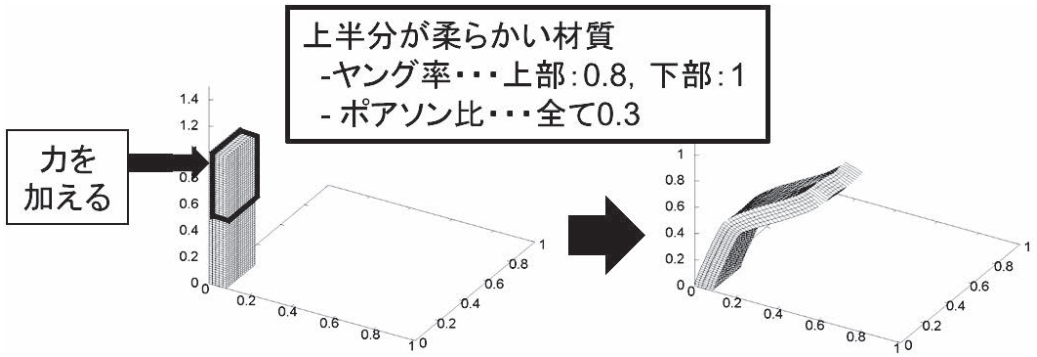

図 5　予備実験での問題設定

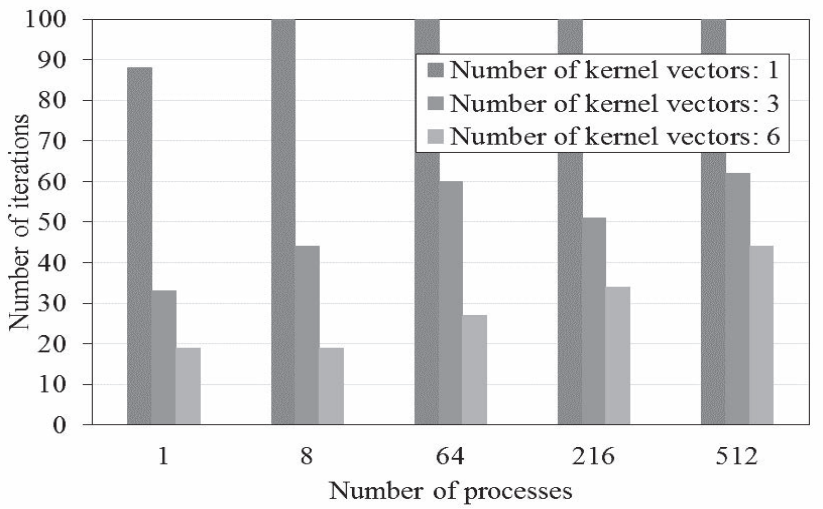

図 6　ニアカーネルベクトルを複数本設定することによる SA-AMG 法の反復回数の変化

４．本研究でのニアカーネルベクトル抽出手法

この節では，本研究で用いたニアカーネルベクトルを問題行列から抽出する手法について説明する．

ニアカーネルベクトルを抽出する手法として，αSA 法と呼ばれる手法が提案されている[6]．αSA 法について説明したものを図 7 に示す．この手法では図 7 のように，ニアカーネルベクトルの条件である$\boldsymbol{Av} \approx 0$を V-cycle で階層的に解き，その結果として出てきたベクトル$\boldsymbol{v}$を補間し，ニアカーネルベクトルとする手法である．本研究では，αSA 法のレベル 1 だけを適用し，ニアカーネルベクトルの抽出を行った．具体的な流れとしては以下のようになる．

1. $\boldsymbol{v}$を乱数で初期化.
2. $\boldsymbol{Av} \approx 0$を V-cycle で μ 回適用し解く.
3. 2. で出てきた解$\boldsymbol{v}$をニアカーネルベクトルとして登録．抽出したい本数に達していない場合，1. に戻る.
4. 抽出したニアカーネルベクトルを出力し，終了.

$\boldsymbol{Av} \approx 0$をV-cycleで解くことにより,V-cycleで解ききれなかった収束しにくい成分が残る．これをカーネルベクトルとして指定し，さらに V-cycle で解くことにより，設定したニアカーネルベクトルと独立したニアカーネルベクトルが抽出できることが考えられる．これにより，適切なニアカーネルベクトルが複数本抽出できると予想できる．この手法では，最初にニアカーネルベクトルを設定する必要があり，このベクトルをもとに独立したニアカーネルベクトルを抽出していくこととなる．

2. でパラメータとしてμが存在しているが，これは外部から入力する値となっている．この値は，V-cycle を繰り返す回数であるため，この値により抽出したニアカーネルベクトルの性質が決まると考えられる．これより，μにより性能が変化することが考えられるが，このことについては今後の課題とし，本研究ではμを 20 で固定し，計測を行っている．

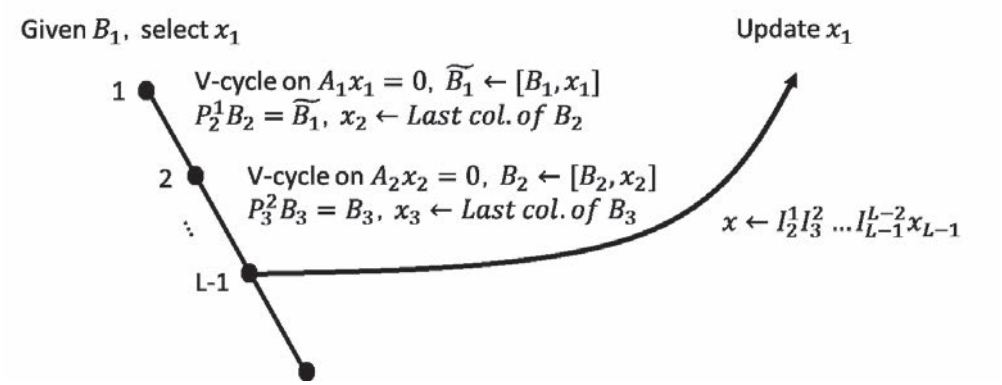

図 7　αSA 法の概略

５．数値実験

５．１　実験環境と使用した問題

本研究では，東京大学の Fujitsu PRIMEHPC FX10 スーパーコンピューターシステム（Oakleaf-FX）を使用し，数値実験を行った[7]．FX10 は，1 ノードに 1 個の SPARC64IXfx プロセッサ（16cores，1.848GHz）と 32GB メモリ（85GB/sec.）を搭載している．また，FX10 では 6 次元メッシュ/トーラス構成（5GB/s/link，bidirectional）を採用している．

数値実験では，最大 512 コアを使用し，計測を行った．また並列化手法については，1 コアに 1 プロセス起動するフラット MPI を使用した．

また本研究では，3 次元弾性体の問題を使用した．この問題は，ある物体に対して，力を加えたとき，どれだけ物体が変形するかを解く問題となっている．またこの問題は 3 次元なため，行列に落とし込む際には 1 節点あたりの行列要素が 3×3 のブロック行列となる．実験 2 で用いた弾性体の問題設定を図 8 に示す．この弾性体の問題では図 8 のように，中に硬い材質が入っている立方体の物体に対して，ある一部分に力を加えたとき，どのように変形するかを解く問題となっている．また，弾性体の問題において硬さを表すヤング率を，硬い材質の部分では 5 に，そのほかの部分を 0.5 に設定している．ヤング率は値が大きければ物体が硬いことを表している．問題サイズについては，ウィークスケーリングで計測しており，1 プロセスあたり 15×15×15 となるように計測を行った．ウィークスケーリングは，1 プロセスが担当する問題サイズが一定になるように問題サイズを設定するようにしたものである．そのため，プロセス数が増加するほど全体の問題サイズが増加する．また，問題行列の各プロセスへの分割は，各軸方向へ問題を均等に分割し，それにより作成された小行列を各プロセスへ分配する単純な方法で分割を行った．

数値実験では AMGS ライブラリ[8]を使用している．AMGS ライブラリは，大規模な疎行列係数の線型方程式を AMG 法で解くライブラリである．解法部では，GPBiCG 法[9]を使用し，前処理として AMG 法を適用している．AMG 法における解法部の緩和法には，対称ガウス・ザイデル法を適用する．そして，解法部の V-cycle の各レベルで緩和法を 2 回適用する．ただし，領域境界では，依存関係を無視している．また節点数が 100 以下になったとき，最も粗いレベルとし，LU 分解を行い，解を求めている．また，反復の終了条件は相対残差が1.0×10^{-7}とし，反復回数が 500 回となったときに，収束しなかったとした．

AMGS ライブラリと解法部の GPBiCG 法は，Fortran を用い作成を行っている．また，コンパイラは Fujitsu Fortran コンパイラ，コンパイラオプションは高速化のための最適化オプションである”-Kfast”と，OpenMP を用いるためのオプションである”-Kopenmp”を使用した．

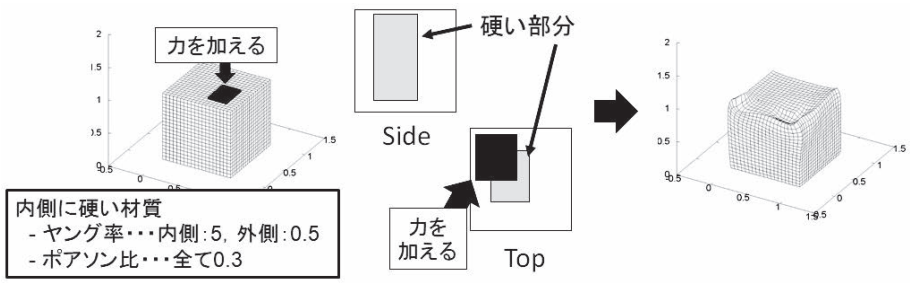

図 8　実験で使用した 3 次元弾性体問題の問題設定

５．２　実験結果

本研究では，抽出したニアカーネルベクトルの設定本数を変更することによる性能変化の分析，および問題行列から予想できるニアカーネルベクトルを設定したときの性能との比較を行い，ニアカーネルベクトル抽出手法の有用性の検証を行った．比較対象を表 1 に示す．表 1 の平行移動成分と回転成分は，弾性体の問題においてニアカーネルベクトルとして知られているベクトルである．また，ニアカーネルベクトルを抽出する際に，平行移動成分をニアカーネルベクトルとして登録してから抽出を行っている．そのため，実際に解く際にも平行移動成分に抽出したニアカーネルベクトルを加え、計測を行っている．

本実験の実行結果を図 9 に示す．図 9 の 5 つのグラフは，それぞれ 1，8，64，216，512 並列で計測した時の結果を示している．これらのグラフの横軸は使用したニアカーネルベクトルを示している．また，図 9 の棒グラフは構築部（Setup part）と解法部（Solve part）の実行時間を示し，折れ線グラフは反復回数を示している．棒グラフの縦軸は実行時間であり，下にあるほど実行時間が少なく，性能が良いということを表している．また，折れ線グラフの縦軸は反復回数であり，下にあるほど反復回数が少なく，収束性が良いということを表している．さらに，反復回数の上限である 500 回に達したものは，反復回数と実行時間ともに空欄としている．また今回は，ニアカーネルベクトルの抽出にかかった時間に関しては着目せず，記載していない．図 9 より，平行移動成分のみ設定した 3p と比べ，平行移動成分と回転成分を設定した 6p が反復回数と実行時間ともに改善がみられることがわかる．さらに，3p や 6p よりも，抽出したニアカーネルベクトルを適切な本数用いることにより，反復回数と実行時間双方で改善がみられることがわかる．しかし，抽出したカーネルベクトルを多く用いれば良いわけではないこともわかる．例として，512 並列の箇所を見てみると，反復回数と実行時間双方で最も良い性能であったニアカーネルベクトルの設定は 3p+3 である．しかし，最もニアカーネルベクトルの設定本数が多い 3p+7 の場合は，反復回数が上限に達してしまい，解けないという結果となった．また，64 並列に着目すると，反復回数で最も良い結果となった箇所は 6p であるのに対し，実行時間が最も良い結果となったのは 3p+1 となった．これは，6p は 6 本のニアカーネルベクトルを用いているが，3p+1 は 4 本のニアカーネルベクトルを用いており，6p の計算量が多く，これが実行時間に現れたものと考えられる．

次に，図 9 においての最良値を示したグラフを図 10 に示す．図 10 は左のグラフが反復回数，右のグラフが実行時間となっている．左のグラフの横軸はプロセス数，縦軸が反復回数となっており，下に行くほど反復回数が少なく，収束性が良いことを表している．また，右のグラフは横軸がプロセス数，縦軸が実行時間となっており，同じく下に行くほど実行時間が短く性能が良いことを示している．さらに，グラフの要素である“3 provided” は平行移動成分のみを設定したときの結果，“6 provided” は平行移動成分と回転成分を設定したもの，“Best in extracted vectors” は抽出したニアカーネルベクトルの中で，最も良い性能を示したときの本数を設定したときの結果を示している．このグラフより，抽出したニアカーネルベクトルを適切な本数設定することで，反復回数と実行時間双方で改善がみられることがわかる．例えば，今回用いた最大のプロセス数である 512 並列に着目すると，抽出したニアカーネルベクトルを適切な本数設定したもの（Best in extracted vectors）は，平行・回転成分を設定したもの（6 provided）に比べ，反復回数が約半分に，実行時間も約 40% 削減されていることがわかる．しかし，並列度および問題サイズを増加させると（ウィークスケーリングなため，並列度を増や

すことと問題サイズを増加させることは同義である)，全体で反復回数が大きく増加し，それにつれて実行時間も大きく増加してしまっていることもわかる（例えば“Best in extracted vectors”では，反復回数が 64 並列は 47 回であるが，512 並列では 141 回と 3 倍に増加しており，その他の要素に関してはそれ以上となる)．

並列度および問題サイズの増加により反復回数が増加してしまう要因として，我々は問題設定が難しいためではないかと考えた．そこで，問題の設定を簡単なものに変更し，再度計測を行った．この実験の結果を図 11 に示す．この計測では問題設定を簡単にするために，図 8 でのヤング率の設定を均一に 1 に設定した．また，このグラフの横軸はプロセス数，縦軸は反復回数となっており，下にあるほど反復回数が少なく，収束性が良いことを示している．このグラフより，簡単な問題であれば反復回数が問題サイズによらず，ほぼ一定になることがわかる（例えば，“Best in extracted vectors” では，64 並列は 8 回，512 並列では 12 回と 1.5 倍となっており，ほかの要素の増加率も同程度である)．

上記の結果より，抽出したニアカーネルベクトルを適切な本数設定することにより，性能を改善することができることがわかった．ここで，抽出したニアカーネルベクトルの適切な設定本数の，検証にかかる時間に着目する．図 12 は，抽出したニアカーネルベクトルにおいて，最良の設定本数の検証にかかる時間のグラフである．このグラフの横軸は実行時間となっており，縦軸は使用したプロセス数となっている．また抽出したニアカーネルベクトルの中で最良の設定本数を検証するためには，現状では抽出にかかる時間と，抽出したニアカーネルベクトルすべてを実行する時間の双方が必要となる（今回の場合，7 本抽出する時間に加え，3p+1，3p+2，…とすべて実行する時間が必要となる)．そのため，図 12 は 7 本のニアカーネルベクトルの抽出にかかった時間と，3p+1 から 3p+7 までの実行が完了するのにかかった時間の合計をグラフに示している．グラフの要素である“Extract time” は 7 本のニアカーネルベクトルの抽出にかかった時間の合計を示し，“Solution time” は 3p+1 から 3p+7 の実行にかかった時間の合計を示している．図 9 と図 12 より，216 並列においてのニアカーネルベクトル検証時間は，6p と比べ約 9 倍の時間がかかっていることがわかる(ニアカーネルベクトルの検証時間は約 300 秒，6p の実行時間は約 34 秒である)．これより，今回我々が用いたニアカーネルベクトル抽出手法は，同じ問題行列を対象に，右辺ベクトルを変えるなどをして 9 回以上解くような場合に有用であることがわかった．また図 12 において，プロセス数が増加すると，それにつれて検証時間が増えてしまっていることもわかる．これはウィークスケーリングであるため全体の問題サイズがプロセス数につれて大きくなり，通信時間が増加してしまっていることや，問題サイズ増加により反復回数が増加してしまっていることが原因であると考えられる．

表 1　本研究の比較対象

ニアカーネルベクトル	詳細
3 provided (3p)	各軸方向 (X, Y, Z) への平行移動成分 (3 本)
6 provided (6p)	平行移動 (3 本) ＋各軸方向への回転成分 (3 本)
3p+1, 3p+2, …	平行移動 (3 本) ＋抽出したニアカーネルベクトル (最大 7 本)

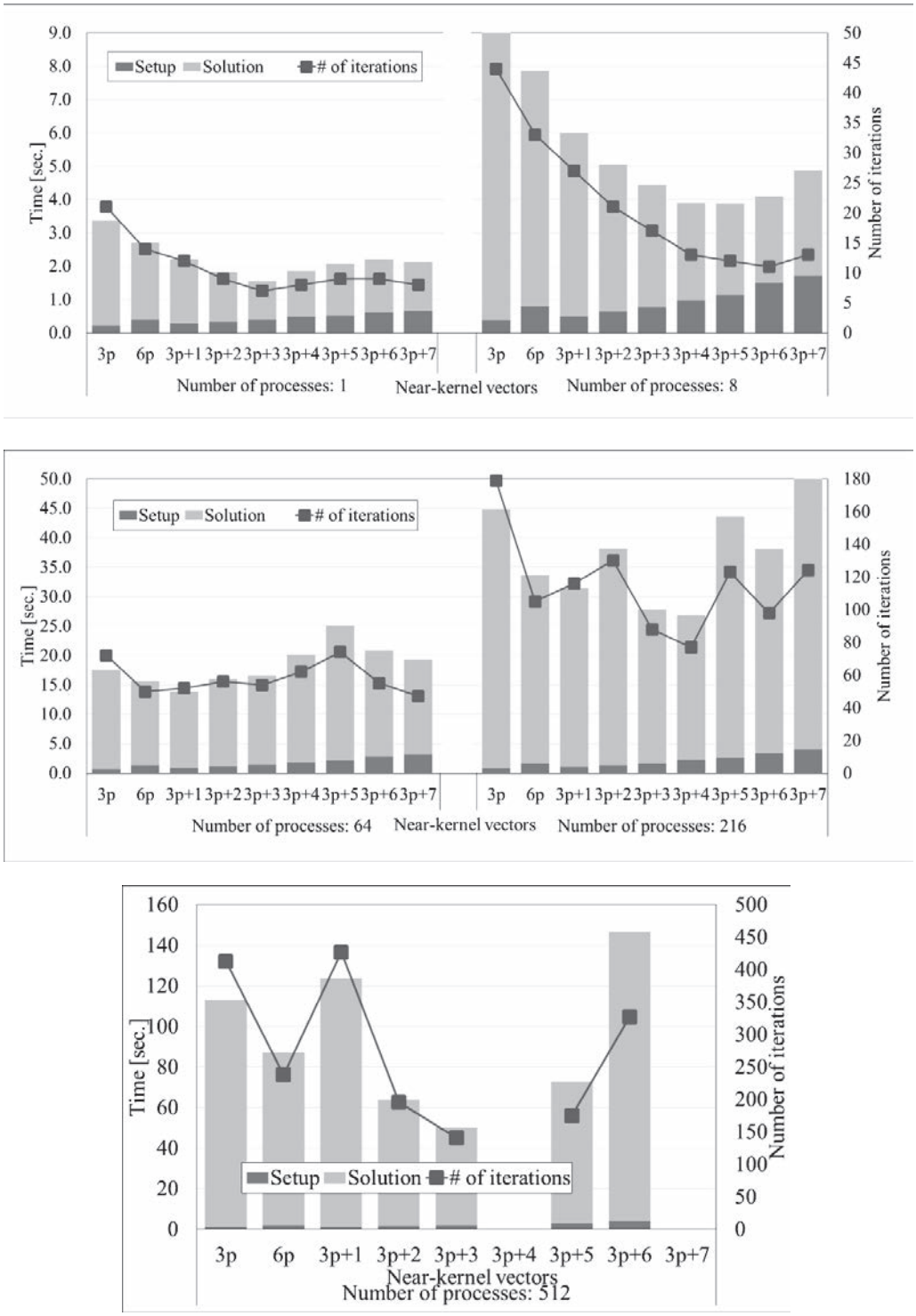

図 9　様々なニアカーネルベクトルを設定したときの実行結果

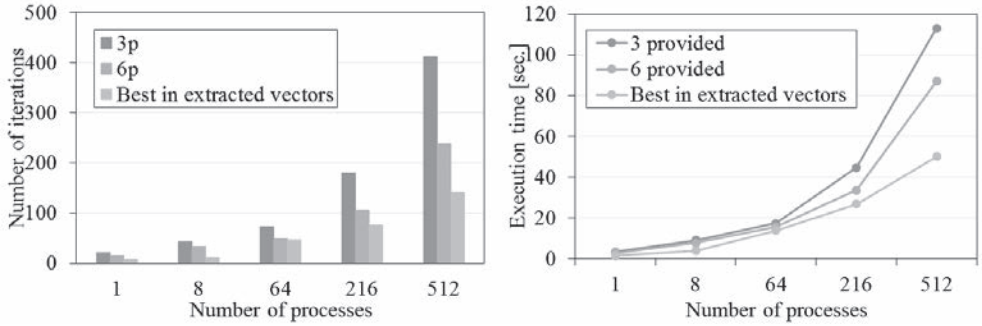

図 10　ニアカーネルベクトルの設定を変更したときの反復回数（左）と実行時間（右）の比較（3 provided と 6 provided，さらに抽出したニアカーネルベクトルのなかで最良だったもの（Best in extracted vectors）　）

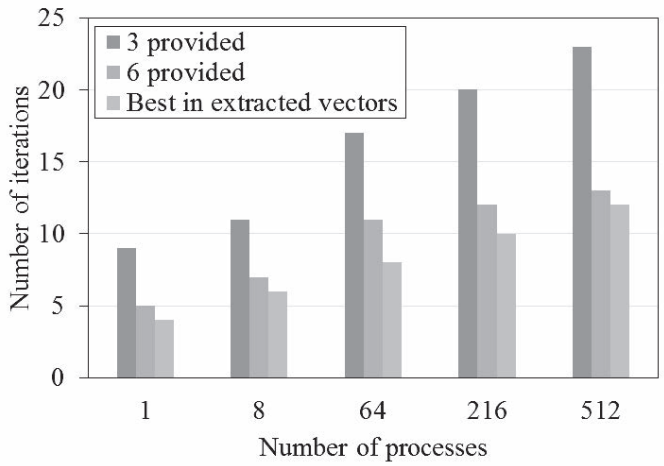

図 11　簡単な問題において，ニアカーネルベクトルの設定を変更した時の反復回数（ヤング率を 1:1 に変更）

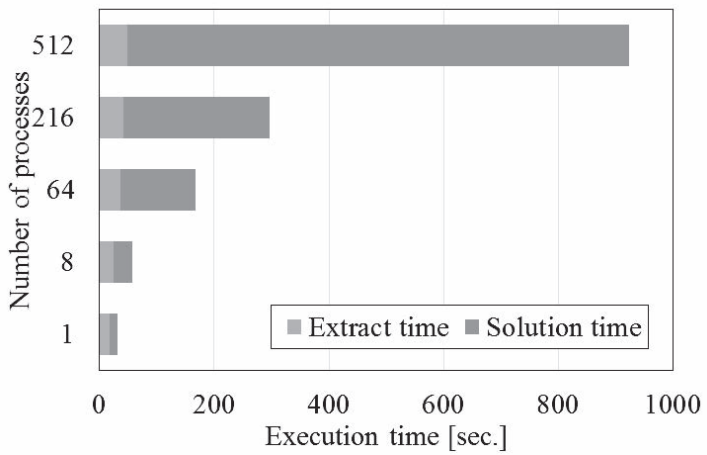

図 12　抽出したニアカーネルベクトルの中で最良の設定本数を見つけるための検証時間

６．結論

本論文では，V-cycle を用いて問題行列からニアカーネルベクトルを複数本抽出する手法を実装し，それらを SA-AMG 法に用いることによる収束性と実行時間への影響を分析した．この実験では 3 次元弾性体の問題を使用しており，この問題でニアカーネルベクトルとして知られている平行移動成分と回転成分を設定するものと，抽出したニアカーネルベクトルを複数本設定する手法で比較も行った．この実験により，平行移動成分のみ設定するよりも，平行移動成分と回転成分を用いることで,反復回数と実行時間双方で改善がみられることがわかった.また，平行移動成分と回転成分を設定するものよりも，抽出したニアカーネルベクトルを適切な本数することで，さらに反復回数と実行時間双方で改善がみられることがわかった．特に，512 並列のときに，抽出したニアカーネルベクトルを適切な本数設定したものは，平行・回転成分を設定したものと比べ,反復回数が約半分となり,実行時間も約 40% 削減されることがわかった．一方で，抽出したニアカーネルベクトルを多く設定すればよいわけではないこともわかった．例えば，512 並列のときに最も良かったものは 3p+3 であり，最もニアカーネルベクトルの使用本数が多い 3p+7 は収束せずという結果となった．

今後の課題としては，まず抽出したニアカーネルベクトルの分析を行うことがあげられる．抽出したニアカーネルベクトルを多く用いることが必ずしも性能改善につながらないことから，SA-AMG 法に設定するものとしては性質の良くないニアカーネルベクトルが抽出されてしまっていることが考えられる．これに関しては，抽出時の残差履歴や，実際に解法を数反復適用させ，性質を見極めることができるかといったことの調査を行うことを考えている．

参 考 文 献

[1] Pereira, F. H., Verardi, S. L. L., and Nabeta, S. I.: “A fast algebraic multigrid preconditioned conjugate gradient solver”, Applied Mathematics and Computation 179, pp.344-351, (2006).

[2] Vanek, P., Brezina, M. and Mandel, J.: “Convergence of Algebraic Multigrid Based on Smoothed Aggregation”, Numerische Mathematik 2001, vol 88, pp.559-579, (2001).

[3] Vanek, P., Mandel, J. and Brezina, M.: “Algebraic Multigrid by Smoothed Aggregation for Second and Fourth Order Elliptic Problems”, Computing, Vol.56, pp.179-196, (1998).

[4] Chan, T. F. and Vanek, P.: “Multilevel algebraic Elliptic Solvers”, UCLA Math, Dept.CAM Report, (1999).

[5] Brezina, M., Falgout, R., Maclachlan, S., Manteuffel, T., Mccormick, S. and Ruge, J.: “Adaptive Smoothed Aggregation (αSA)”, SIAM J. Sci. Comput, Vol. 25, No.6, pp.1896-1920 (2004).

[6] 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).

[7] Information Technology Center: The University of Tokyo, http://www.cc.u-tokyo.ac.jp/.

[8] AMGS Library: http://hpcl.info.kogakuin.ac.jp/lab/software/amgs.

[9] Zhang, S.-L.: “GPBi-CG: Generalized Product-type Methods Based on Bi-CG for Solving Nonsymmetric Linear Systems”, SIAM J. Sci. Comput, Vol. 18, No.2, pp.537-551 (1997).