情報処理学会論文誌　コンピューティングシステム　Vol.12　No.3　46–63 (July 2019)

高スケーラブル・安定的なSA-AMG法に向けたニアカーネルベクトル自動抽出手法に関する研究

野村 直也[1,a)]　中島 研吾[1,b)]　藤井 昭宏[2,c)]

受付日 2018年12月4日, 採録日 2019年3月20日

概要：本論文では，マルチレベルな解法である多重格子（Multigrid）法の 1 種である SA-AMG（Smoothed Aggregation-Algebraic MG）法の，一般の問題に対する高スケーラブル・安定化に向け，粗いレベルでニアカーネルベクトルを抽出，さらに適切な抽出本数を予測する手法を提案する．Multigrid 法は，高速で安定な収束性やスケーラブル性を持つ解法として広く知られているが，悪条件の問題においては収束性が悪化する傾向にあることが知られている．著者らの先行研究などから，SA-AMG 法において粗い問題生成時に，問題に応じたニアカーネルベクトル $\boldsymbol{e}$（$A\boldsymbol{e} \approx 0$ となる非ゼロベクトル）の設定により，より高速で安定な収束が実現できることが分かっている．このベクトルに関して，αSA 法や著者らによる先行研究がある．αSA 法は，全階層の問題行列を網羅するようなニアカーネルベクトルを，1 本ずつ抽出する手法である．著者らは 1 本のベクトルのみに基づいて，全階層を完全に網羅することは難しいと考えた．そこで先行研究では，与えられた問題行列が置かれる 1 階層目のみ抽出を行う手法を提案した．数値実験により，適切な本数のニアカーネルベクトルを用いることで，先行研究の手法は有用となることを示した．本論文では新たな抽出手法として，粗いレベルのニアカーネルベクトルは，独立に粗いレベルで新たに複数本抽出および追加し，さらに抽出時点で適切な抽出本数を予測する手法を提案する．数値実験により，用途に応じて適切なパラメータを設定することで，本手法が従来手法と比べ，有用となることが分かった．

キーワード：Multigrid 法，ニアカーネルベクトル抽出手法

The Study of the Automatic Extraction Method of Near-kernel Vector for High Scalable and Stable SA-AMG Method

Naoya Nomura[1,a)]　Kengo Nakajima[1,b)]　Akihiro Fujii[2,c)]

Received: December 4, 2018, Accepted: March 20, 2019

Abstract: In this paper, we propose the extraction method of near-kernel vectors at coarser levels and the prediction method of the optimum extraction number for high scalable and stable SA-AMG method at general problem. SA-AMG method is one of the Multigrid method. Multigrid method is known as the method which has fast and stable convergence and scalability. But, the convergence tends to deteriorate in bad condition problem. From our previous study, by using optimum near-kernel vectors when the multilevel matrices are made, the convergence of SA-AMG becomes good. There are some study about near-kernel vectors, like αSA and our previous study. αSA is the extraction method of near-kernel vectors that covers all level's matrices. We consider that it is difficult to cover all level's matrices completely. Therefore, in our previous study, we propose the extraction method of near-kernel vectors at only level 1 where the original matrix is placed. By numerical experiments, this method is good by setting optimum number of near-kernel vectors. In this paper, we propose new extraction method that extracts at coarser levels and predict the optimum number of near-kernel vectors. By numerical experiments, our proposed method is good by setting optimum parameters according to the application.

Keywords: Multigrid method, the extraction method of near-kernel vectors

[1] 東京大学
The University of Tokyo, Bunkyo, Tokyo 113–8656, Japan
[2] 工学院大学
Kogakuin University, Shinjuku, Tokyo 163–8677, Japan

[a)] naoya_nomura@mist.i.u-tokyo.ac.jp
[b)] nakajima@cc.u-tokyo.ac.jp
[c)] fujii@cc.kogakuin.ac.jp

1. はじめに

有限要素法，差分法のようなコンピュータによるシミュレーションに基づく計算科学は，最終的に連立 1 次方程式 $\boldsymbol{Ax} = \boldsymbol{b}$ を解くことに帰着されることが多い．近年では，より複雑な問題を正確にシミュレートすることが求められており，それにより問題の大規模化が進んでいる．そのため，大規模連立 1 次方程式を高速かつ安定に解く手法の開発は急務となっている．

そこで，大規模連立 1 次方程式を高速に解く手法として Multigrid 法が使用されている．Multigrid 法は，与えられた問題行列から粗い問題を再帰的に生成する問題生成部（以下，構築部）と，それらを用いて問題を解く反復解法部（以下，解法部）に分かれている．これにより，Multigrid 法は様々な波長の誤差成分を効率良く減衰させ，高速で安定な収束やスケーラビリティを実現している．Multigrid 法は大きく分けて，幾何的 Multigrid（GMG）法と，代数的 Multigrid（AMG）法 [1] に分かれており，本研究では AMG 法を対象とする．AMG 法の中でも階層行列の生成方法により，さらに様々な派生解法が存在する [2]．本研究では特に，Smoothed Aggregation に基づく AMG（SA-AMG）法と呼ばれる手法を対象とした [3], [4], [5]．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．

SA-AMG 法における構築部では，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．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を用い，Gauss-Seidel 法などの緩和法を用いて問題行列を解く．このとき，Multigrid 法では主に，V-cycle と呼ばれる手法を用いて階層的に解いていく（詳細は 2.2.2 項で述べる）．元の問題行列などの大規模な問題が設置される階層（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，収束しにくい誤差成分を粗いレベルで効率良く減衰させることで，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，問題行列 A との行列ベクトル積が 0 に近くなるような非ゼロベクトルをいう．Gauss-Seidel 法のような通常の定常反復解法で解いた際，この成分が収束の停滞を引き起こす要因となることが知られている．SA-AMG 法ではこのような誤差成分を効率良く減衰させるため，構築部においてニアカーネルベクトルを用いて粗いレベルの行列を生成する．これにより，粗いレベルの行列に誤差成分が射影され，誤差成分を効率良く減衰させることが可能となる．その結果，残差を効率良く収束させることができる．

本研究の手法は一般の問題を対象としている．本研究ではその中で，3 次元弾性体の問題を用いて実験を行い，本研究の手法の有用性を検証した．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（詳細は 3.1 節で述べる）．著者らによる先行研究により，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双方で改善がみられることが分かっている（3.2 節で示す）．しかし，これは問題設定に依存したものであり，すべての問題において適用できるものではない．すべてのユーザが，対象とする問題の物理的性質に基づいた，適切なニアカーネルベクトルの設定を行うことができるとは限らない．SA-AMG 法の高い収束性をより多くの問題に対して生かすために，係数行列 A から適切な数のニアカーネルベクトルを，代数的に効率良く抽出する手法の開発が急務である．そこで著者らは，問題行列に応じた適切なニアカーネルベクトルの設定方法と，その効果についての研究を行ってきた．著者らの現在までの研究では，αSA 法 [6] と呼ばれる手法を基に，ニアカーネルベクトル抽出の効率化のため，V-cycle を用いてレベル 1（元の問題行列が置かれる最も細かいレベル）のみにおいて，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の提案，および有用性の検証を行った（詳細は 4 章で述べる）．そして，その手法で抽出したニアカーネルベクトルを SA-AMG 法に適切な本数設定する（適切な本数の設定については，4.2.3 項で実験結果を交え，詳細を述べる）ことで，平行移動成分と回転成分を設定した場合よりも，反復回数と実行時間双方で改善がみられることが分かった [7]．

文献 [6] や [7] を含め，従来の関連研究では，階層の全レベルにおいて同一本数のニアカーネルベクトルを用いて設定を行っている（たとえば，細かいレベルで 3 本設定した場合，粗いレベルでも 3 本用いて計算を行っている）．これは SA-AMG 法では通常，細かいレベルのニアカーネルベクトルをもとに，次の粗いレベルのニアカーネルベクトルを生成しているためである．しかし著者らは，細かいレベルから生成されたニアカーネルベクトルだけでは，粗いレベルの行列におけるニアカーネルベクトル成分の減衰が不十分であり，高い収束性を生かしきれていないのではないかと考えた．また文献 [7] において，ニアカーネルベクトルを「適切な」本数設定することができれば，改善がみられることが分かった．しかし文献 [7] でさらに，多く設定することで必ずしも改善するとは限らないことも分かった．そのため，適切なニアカーネルベクトル設定本数の検証が必要であるが，従来では抽出本数のとりうるすべてのパターンを網羅する必要があり，検証に膨大な時間を必要としてしまうという問題があった．そのため，より高い実行効率を得るためには，適切な本数を抽出時点で判明させることが理想である．

本研究の最終目標は，SA-AMG 法の適用性（Applicabil-

ity）の拡張である（適用性について，本研究では，SA-AMG 法の適用可能な問題の範囲とする）．上記で述べたように，SA-AMG 法の高い収束性を多くの問題で生かすためには，適切な数のニアカーネルベクトルを，効率良く抽出する手法の開発が急務である．そこで本研究では文献 [7] の手法を改良し，粗いレベルでニアカーネルベクトルを複数本抽出し，抽出時において適切な設定本数を予測する手法の提案を行う．そして，SA-AMG 法において本手法を用いることによる有用性を，従来手法と比較することで検証した．

本論文は 6 つの章にわかれている．まず 2 章で Multigrid 法の概要，および Multigrid 法の派生解法であり，本研究で用いている SA-AMG 法の概要を示す．次に，3 章で，ニアカーネルベクトルについての詳細な説明を行う．その後，4 章において，本研究で用いたニアカーネルベクトル抽出手法の概要，5 章で適切なニアカーネルベクトル設定本数予測手法付き抽出手法の概要および数値実験を交えた有用性の検証をそれぞれ行う．最後に，6 章で本論文のまとめとする．

2. Multigrid 法

2.1 Multigrid 法の概要

有限要素法のような格子を用いて離散化した問題に対し，Gauss-Seidel 法のような定常反復解法を適用すると，メッシュサイズと同じサイズの誤差が早く減衰し（空間的高周波成分），長い波長の誤差は減衰しにくい（空間的低周波成分）ことが知られている．そこで，Multigrid 法ではこの性質に着目し，粗い格子を階層的に生成し，それらを用いることで，低周波成分を効率良く減衰させ，高速で安定な収束やスケーラビリティを実現している．Multigrid 法の大きな特徴として，収束性が問題サイズによらないスケーラブル性があり，大規模問題に対して非常に有効な解法となっている．Multigrid 法にはさらに，空間離散化の情報に基づき粗い格子を生成する幾何的マルチグリッド（Geometric Multigrid，GMG）法と，一般の連立 1 次方程式 $A\boldsymbol{x} = \boldsymbol{b}$ の係数行列 A の情報のみに基づき粗い格子を生成する，代数的考察に基づいた代数的マルチグリッド（Algebraic Multigrid，AMG）法が存在する．GMG 法は粗い格子の生成に空間離散化の情報が必要となる．一方，AMG 法は係数行列 A のみでよく，不規則メッシュ状の問題やメッシュそのものが存在しない問題まで適用できるため，一般に AMG 法のほうが対象とする問題の適用可能範囲が広い．そのため本研究では，AMG 法を対象とし研究を行っている．AMG 法の中でも，粗い格子の生成方法により，さらに様々な解法が存在する [2]．本研究では，その中で特に SA-AMG 法に着目して研究を行っている [3], [4], [5]．次節より，AMG 法についてより詳細な説明を行い，次に SA-AMG 法についての説明を行う．

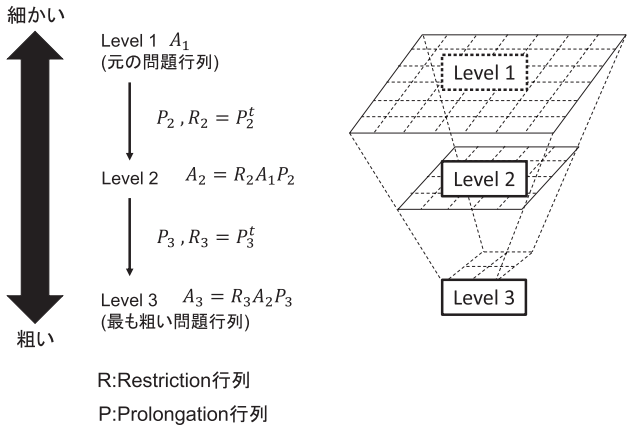

図 1　構築部の概要
Fig. 1　Setup part.

2.2 AMG 法

AMG 法は上記でも述べたとおり，一般の連立 1 次方程式 $A\boldsymbol{x} = \boldsymbol{b}$ の係数行列 A の情報のみに基づき粗い格子を生成する Multigrid 法である．Multigrid 法では一般に大きく分けて，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（構築部），これらを用いて問題行列を解く（解法部），2 つの処理からなる．以下の 2.2.1 項と 2.2.2 項において，AMG 法における構築部と解法部の流れを説明する．

2.2.1 構築部

AMG 法の構築部の概要を**図 1** に示す．構築部では細かいレベルの問題行列を基に，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，粗いレベルの問題行列や，階層移動の際に必要となる補間演算子 Prolongation（P）行列と Restriction（R）行列を生成する．これを再帰的に行うことで，階層的に行列を生成する．図 1 のように，1 レベル目は与えられた問題行列が置かれ，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．AMG 法では一般に，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）を行うことで作成する．また一般に，$R = P^t$ と設定する．

2.2.2 解法部

次に，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つ．この図のように，最上層に与えられた問題行列（A_1）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（$A_2, A_3, \ldots$）が設置される．階層数は問題サイズによって可変となる．解法部の計算を Algorithm 1 に示す．

階層移動（Coarse grid correction の箇所）では，構築部で作成された補間演算子の 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

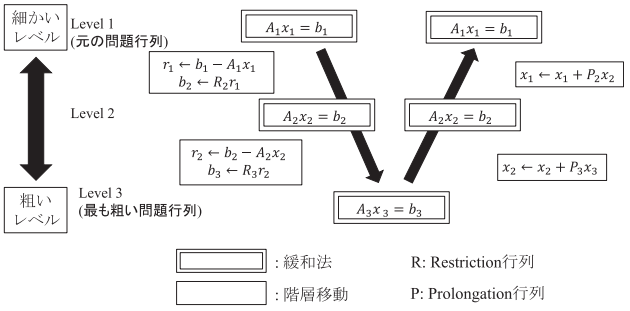

図 2　解法部の概要（V-cycle）
Fig. 2　Solution part (V-cycle).

Algorithm 1 解法部（V-cycle）

Pre-smoothing: $\tilde{\boldsymbol{x}}_{\boldsymbol{l}} \leftarrow S_l(\boldsymbol{x_l}, \boldsymbol{b_l})$
Coarse grid correction:
$\boldsymbol{r_l} \leftarrow \boldsymbol{b_l} - A_l\tilde{\boldsymbol{x}}_{\boldsymbol{l}}$
$\boldsymbol{b_{l+1}} \leftarrow R_l\boldsymbol{r_l}$
if $l+1 = L$ **then**
　$A_L\boldsymbol{x_L} = \boldsymbol{b_L}$ を直接法（例；LU 分解）で解く．
else
　$l+1$ において $\boldsymbol{x_{l+1}} = 0$ とし Pre-smoothing から計算．
end if
$\boldsymbol{x_l} \leftarrow \boldsymbol{x_l} + P_l\boldsymbol{x_{l+1}}$
Post-smoothing: $\boldsymbol{x_{l+1}} \leftarrow S_l(\boldsymbol{x_l}, \boldsymbol{b_l})$

$S(\boldsymbol{x}, \boldsymbol{b})$: $A\boldsymbol{x} = \boldsymbol{b}$ に対するスムーザの適用
$l = 1, 2, \ldots, L$: レベル

いベクトルを生成し，1 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，1 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，Multigrid 法では補間演算子の生成方法は重要となる．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 [2]．AMG 法の代表的な手法の 1 つとして，Stuben [8] や，Brandt ら [9] による，classical な AMG 法がある．この手法では，あらかじめ粗いレベルに移動させる節点集合 C と，C 以外の節点集合 F に分割する．そして，それらの集合を基に集約を行う．また別の手法として，本研究で対象としている SA-AMG 法がある．この手法では，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どうしで集合を作り，その集合内で重み付けをして補間演算子を生成する．その後，生成された補間演算子を基に，集約を行う．SA-AMG 法は様々な分野で利用されており，AMG 法の代表的な手法の 1 つとなっている．

2.3　SA-AMG 法 [3], [4], [5]

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，

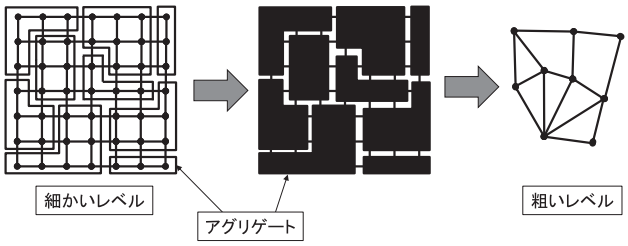

図 3　アグリゲート生成の概要
Fig. 3　How an aggregate is made.

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．アグリゲート生成は具体的に，以下の式によって生成される．

$$N_i(\epsilon) = \{j : |A_{ij}| \geq \epsilon\sqrt{|A_{ii}||A_{jj}|}\} \cup \{i\} \qquad (1)$$

ここで，N_i は i 番目のアグリゲート，A_{ij} は行列 A の i 行 j 番目の要素を示す．この式に示すとおり，i 番目のアグリゲートは i 番目の要素に対応する節点かつ，絶対値が対角成分に対し比較的大きい非対角成分の要素に対応する節点で構成される．またそのうえで，補間の関係上すべての要素がどこかしらのアグリゲートに属していなければいけない．そのため，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，アグリゲートを生成する．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．著者らの研究から，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られることが分かっている．このことについては，次章の 3.1 節および 3.2 節で詳しく述べる．

3.　ニアカーネルベクトル

3.1　ニアカーネルベクトルの概要

ニアカーネルベクトルとは，$A\boldsymbol{e} \approx 0$ となる非ゼロベクトル $\boldsymbol{e}$ をいう（代数的に滑らかな成分とも呼ばれる）．2.1 節で述べたとおり，Gauss-Seidel 法のような定常反復解法を数回適用すると，高周波成分が早く減衰し，低周波成分だけが残るという現象がある．つまり，定常反復解法の反復行列を S と定義すると，$S\boldsymbol{e'} \approx \boldsymbol{e'}$ となる低周波成分 $\boldsymbol{e'}$ が存在する．これにより，定常反復解法において一定回数の反復の後は，収束性が悪化することが多く見受けられる．この成分 $\boldsymbol{e'}$ はニアカーネルベクトル $\boldsymbol{e}$ と同値である．これはたとえば，定常反復解法として減速 Jacobi 法を用いるとすると $S = I - \omega D^{-1}A$ であり，$S\boldsymbol{e'} \approx \boldsymbol{e'}$ を整理すると $\omega D^{-1}A\boldsymbol{e'} \approx 0$（$\omega \neq 0$）が得られ，両辺に対角要素を示す

行列 D と $1/\omega$ をかけることで，$A\boldsymbol{e'} \approx 0$ が導かれる．これより，$\boldsymbol{e'} = \boldsymbol{e}$（少なくとも $\boldsymbol{e'} \approx \boldsymbol{e}$），つまり低周波成分とニアカーネルベクトルが同値であることが分かる [10], [11]．

AMG 法では，このようなニアカーネルベクトル成分を効率良く減衰させることで，高い収束性を実現している．具体的には，近似解 $\boldsymbol{u}$ にニアカーネル成分 $\boldsymbol{e}$ が含まれるとき，適当な $\boldsymbol{v}$ を選んで新しい近似解 $\tilde{\boldsymbol{u}} = \boldsymbol{u} + P\boldsymbol{v}$ のニアカーネル成分 $\tilde{\boldsymbol{e}} = \boldsymbol{e} + P\boldsymbol{v}$ を 0 にすることを考える．ここで単純化のため，2 レベルの AMG 法を考えると，図 2 の V-cycle の式から，粗いレベルにおける方程式は $A_2\boldsymbol{v} = R\boldsymbol{r}$ と表現できる．ここで，$A_2 = R_2A_1P_2 (R = P^T)$（$A$ は正定値対称行列）と設定されていれば，変分原理より新しい近似解 $\tilde{\boldsymbol{u}}$ のニアカーネル成分 $\tilde{\boldsymbol{e}} = \boldsymbol{e} + P\boldsymbol{v}$ が最小になることが知られている [10], [11]．つまり，以下が成り立つ．

$$||e + Pv||_A = \min_w ||e + Pw||_A \tag{2}$$

これらより，補間演算子である Prolongation 行列（Restriction 行列）を適切に選ぶことで，ニアカーネル成分を効率良く減衰できる．ここで，適切な Prolongation 行列を設定するためには $\tilde{\boldsymbol{e}} = \boldsymbol{e} + P\boldsymbol{v}$ から，$\boldsymbol{e} \in \mathrm{Im}\, P$ となることが必要である．そこで SA-AMG 法では，ニアカーネルベクトルを用いて，Prolongation 行列の作成を行っている．Prolongation 行列の作成に用いるニアカーネルベクトルを 1 次独立かつ適切にとることができれば，Prolongation 行列の各行ベクトルの線形結合によりニアカーネルベクトルを表現することが可能となり，$\boldsymbol{e} \in \mathrm{Im}\, P$ が成り立つ．以上より Multigrid 法の特徴である高い収束性の実現のため，実際に SA-AMG 法を適用する際には，ニアカーネルベクトルの生成および設定方法の確立が必要不可欠となる．

問題の性質からニアカーネルベクトルが特定できる場合もある [12]．たとえば，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．弾性体問題は，物体に力が加えられたときの，物体の変形をシミュレートする問題である．弾性体問題は連立 1 次方程式 $A\boldsymbol{u} = \boldsymbol{f}$ に帰着できる．ここで，$\boldsymbol{u}$ は物体の変位，$\boldsymbol{f}$ は物体に加えられた力を示す．平行移動や回転は物体の変形がともなわないため，物体自体に力が加わらない，つまり右辺ベクトル $\boldsymbol{f}$ が 0 となることが知られている．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．これについては，次節の 3.2 節で，数値実験による結果を示す．

ニアカーネルベクトルの補間演算子への設定方法の概要を示す [3], [4], [5]．レベル l における補間演算子の生成を，Algorithm 2 に示す．

レベル $l+1$ 以降においても同様の操作を行い，階層ごとに補間演算子 $P_{l+1}, P_{l+2}, \ldots$ を生成する．

以上の補間演算子作成の流れを，**図 4** に図示する．図 4 は 2 本のニアカーネルベクトルを用いて，問題 A から 2 つのアグリゲートが作成されたときの，Prolongation 行列の作成方法を図示している．図 4 のように，まずアグリゲートの節点番号に対応したニアカーネルベクトルの列要素を，1 次行列 S に入力する（a）．その後，S に対し QR 分解を行い，算出された行列 Q を仮の補間演算子 $\tilde{P}_l$ に入力する（b）．同時に算出された行列 R は，アグリゲート情報を基に，次レベルのニアカーネルベクトル V_{l+1} の構築に用いられる（c）．最後に，仮の補間演算子 $\tilde{P}_l$ に緩和法を適用する（d）．以上の操作を行うことで，補間演算子 P_l や次レベルのニアカーネルベクトル V_{l+1} が生成される．図 4 の

Algorithm 2 補間演算子生成方法

Step.1 Decomposition and Restriction

ニアカーネルベクトル群 V_l から，生成されたアグリゲート情報を基に，1 次行列 $S_1, S_2, \ldots$ ($S \in \mathbb{R}^{N_l \times N^v}$) を作成する（2.3 節でも述べたように，アグリゲート数は次の節点数 N_{l+1} と同一であることに注意）

for $i = 1$ to N_{l+1} **do**
  for $kernel\_num = 1$ to N^v **do**
    $S_i^{j,kernel\_num} \leftarrow v_l^{j,kernel\_num}\,|_{j \in C_i}$,
  end for
end for

Step.2 QR decomposition

S に対して QR 分解を行い，直交行列 $Q \in \mathbb{R}^{N_l \times N^v}$ と上三角行列 $R \in \mathbb{R}^{N^v \times N^v}$ を生成する．

for $i = 1$ to N_{l+1} **do**
  $\tilde{S}_i \rightarrow Q_iR_i$
end for

その後，Q を基に仮の補間演算子 $\tilde{P}_l \in \mathbb{R}^{N_l \times (N^v N_{l+1})}$ を生成する．

for $i = 1$ to N_{l+1} **do**
  $\tilde{P}_l^{\cdot,(i-1)N^v} \leftarrow Q_i$
end for

次のレベルのニアカーネルベクトル V_{l+1} を，R 行列を基に作成する．$\tilde{P}$ の作成時に使用したアグリゲート情報から，行列 R は次の粗いレベルの節点と対応できる．行列 R を次レベルの節点情報を基に組み合わせ，次レベルのニアカーネルベクトルを作成する．

for $i = 1$ to N_{l+1} **do**
  $v_{l+1}^{(i-1)N^v,\cdot} \leftarrow R_i$
end for

Step.3 Final smoothing

$\tilde{P}$ をこのまま Prolongation 行列として用いてもよいが，最後に要素間の係数を滑らかにするため，スムーザを適用する．本研究では減速ヤコビ法を適用しており，以下のように表記される．

$P_l \leftarrow (I - \omega D^{-1}A_l)\tilde{P}_l$

C_i: i 番目のアグリゲートに含まれる要素集合
N_l: レベル l における未知数個数
N^v: ニアカーネルベクトルの設定本数
$a_l^{i,j}$: レベル l におけるベクトル $\boldsymbol{a}$ の i 行 j 列目の要素（行要素すべてを参照する場合は $a_l^{i,\cdot}$，列要素は $a_l^{\cdot,j}$ と表記）
$P_l = \{p_l^{\cdot,1}, p_l^{\cdot,2}, \ldots\ (p \in \mathbb{R}^{N_l})\}$: レベル l における Prolongation 行列
$V_l = \{v_l^{\cdot,1}, v_l^{\cdot,2}, \ldots\ (v \in \mathbb{R}^{N_l})\}$: レベル l におけるニアカーネルベクトル群の行列
D: A の対角要素
ω: 減速係数

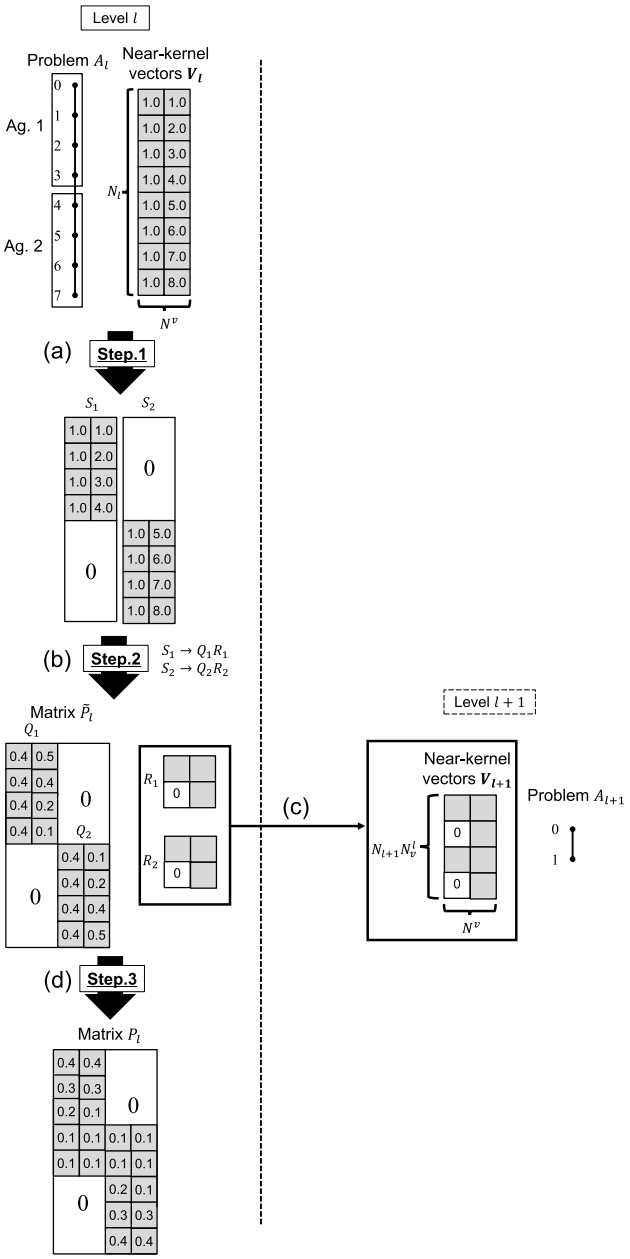

図 4　補間演算子（Prolongation 行列）の生成方法

Fig. 4　The construction of the interpolate matrix (Prolongation matrix).

例では Level$l+1$ において，ニアカーネルベクトルを要素番号と対応させるために，1 要素が 2×2 のブロック行列として扱う必要がある．このように，次レベルにおけるニアカーネルベクトル V_{l+1} は，1 節点が $N_v \times N_v$ のブロック行列として扱うこととなる．図 4 は 2 本のニアカーネルベクトルを用いた例だが，本研究の SA-AMG 法では，より多くのニアカーネルベクトルを設定することができる．その場合，Step.1 における補間演算子の候補行列 $\hat{P}$ の行列サイズが大きくなり，結果として計算量が増加する．そのため，ニアカーネルベクトルを複数本設定することで反復回数が減少するが，1 反復あたりの計算時間が増加するといった，トレードオフが発生すると考えられる．

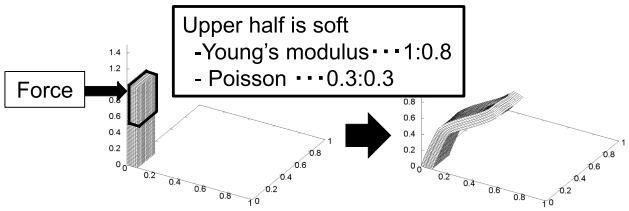

図 5　事前実験における問題設定

Fig. 5　Problem settings at the previous experiment.

表 1　事前実験における比較対象

Table 1　The comparison of the previous experiment.

比較対象	ニアカーネルベクトル設定本数
Case 1	1（要素がすべて 1 の定数ベクトル）
Case 2	3（平行移動成分のみ）
Case 3	6（平行移動 + 回転成分）

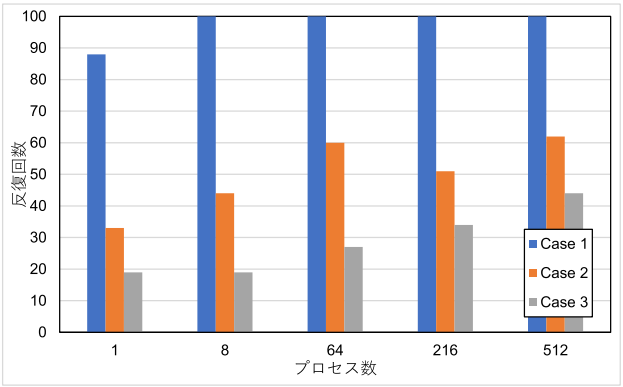

図 6　ニアカーネルベクトル設定による SA-AMG 法の反復回数の変化（事前実験結果）

Fig. 6　The effect of the convergence of SA-AMG method by setting near-kernel vectors (the result of the previous experiment).

3.2　事前実験

3.2.1　SA-AMG 法におけるニアカーネルベクトル設定による収束性検証

図 5 と表 1，および図 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ための，事前実験の問題設定と比較対象，およびその結果を示す．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，4.2.1 項で詳しく説明する．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 $6 \times 15 \times 60$ としたウィークスケーリングで計測を行った．さらに表 1 において，事前実験における比較対象を示す．表 1 に示すように，事前実験ではニアカーネルベクトルの設定本数により，3 種類の比較対象を用意し実験を行っ

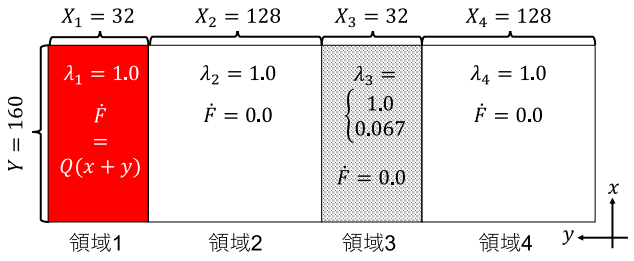

図 7　本研究で用いた 2 次元定常熱伝導問題の問題設定
Fig. 7　The problem settings of 2D heat transfer problem.

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．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分かる．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．本研究では数値実験において，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良いニアカーネルベクトルが設定できるかの検証も行う．

3.2.2　2 次元定常熱伝導問題に対するニアカーネルベクトル設定による収束性検証

前項において，3 次元弾性体問題においてニアカーネルベクトルの複数本設定による，収束性における有用性を検証した．そこで，3 次元弾性体問題以外の問題に対しても，ニアカーネルベクト設定の有用性を示す．本項では，2 次元定常熱伝導問題

$$\frac{\partial}{\partial x}\left(\lambda_{x,y}\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial y}\left(\lambda_{x,y}\frac{\partial u}{\partial y}\right)+\dot{F}(x,y)=0 \qquad (3)$$

（ただし，本実験では 4 面における領域境界 Γ 上で Dirichlet 境界条件（$\Delta u(x,y)=0$（$x,y\in\Gamma$））を付している）に対し，ニアカーネルベクトルを複数本設定した際の有用性を示す．本実験で使用した問題の概要を**図 7** に示す．図 7 のように，本実験の問題では 4 つの領域に分割し，それぞれの領域で異なるパラメータ設定を行った．まず，要素数を $X_1 = X_3 = 32$，$X_2 = X_4 = 128$，$Y = 160$ と設定した．また，領域 1 においての発熱量 $\dot{F}$ を，節点座標 (x,y) に依存するように設定し（本実験では $Q = 1.0$ に設定），その他の領域では $\dot{F} = 0.0$ に設定した．さらに熱伝導率を，領域 1 や領域 2 および領域 4 では $\lambda_1 = \lambda_2 = \lambda_4 = 1.0$ と設定し，領域 3 における熱伝導率 λ_3 を，他領域と同等の $\lambda_3 = 1.0$ と不均質性を持たせた $\lambda_3 = 0.067$ の 2 種類の設定において実験を行った．

本実験の結果を以下に示す．本実験では，Oakforest-PACS スーパーコンピュータシステムにおいて，1 コアのみを用いて実験を行った．また本実験では比較対象として，ポアソン方程式や拡散方程式のニアカーネルベクトルとして知られている定数ベクトル $(1,1,\ldots)^T$ を，ニアカーネルベクトルとして設定した際の結果を 1p として記載している．均質性問題（$\lambda_3 = 1.0$）における結果を**表 2**，不均質性問題（$\lambda_3 = 0.067$）における結果を**表 3** に示す．不均質性が強い問題では条件数が大きくなり，それにともない収束性も悪化する．実際，表 3 の 1p では収束性の悪化が見受けられる．しかし，ニアカーネルベクトルを適切に設定することで，高い収束性を発揮していることが分かる．さらに，ニアカーネルベクトルの設定本数を増加させることで，さらに収束性の改善がみられることも分かる．以上より，ニアカーネルベクトルを適切に設定することで収束性が改善し，特に不均質性が強い，条件数が大きい問題に対して高い効果を発揮することが分かる．4 章と 5 章では，本研究における提案手法の説明および有用性の検証を行うが，数値実験では 3 次元弾性体問題に着目し実験を行う．

表 2　2 次元定常熱伝導問題に対するニアカーネルベクトル設定本数による反復回数の変化（均質性問題，$\lambda_3 = 1.0$）
Table 2　The result of the number of iteration at 3D heat transfer problem (isotropic problem, $\lambda_3 = 1.0$).

	ニアカーネルベクトル設定本数				
	1p	1p+1	1p+2	1p+3	1p+4
レベル 1 のみ	5	4	4	3	3
粗いレベルで抽出：+1	—	4	4	3	3
粗いレベルで抽出：+5	—	4	4	3	3

表 3　2 次元定常熱伝導問題に対するニアカーネルベクトル設定本数による反復回数の変化（不均質性問題，$\lambda_3 = 0.067$）
Table 3　The result of the number of iteration at 3D heat transfer problem (anisotropic problem, $\lambda_3 = 0.067$).

	ニアカーネルベクトル設定本数				
	1p	1p+1	1p+2	1p+3	1p+4
レベル 1 のみ	81	4	4	3	3
粗いレベルで抽出：+1	—	4	4	3	3
粗いレベルで抽出：+5	—	4	4	3	3

3.3　ニアカーネルベクトル抽出手法に関する先行研究

3.3.1　関連研究

3.2 節から，ニアカーネルベクトルの設定が SA-AMG 法の収束性に大きくかかわることが分かる．そのため，どのようなニアカーネルベクトルを設定するかを決めることは，SA-AMG 法において重要なこととなる．通常，このニアカーネルベクトルの設定に関しては，問題の性質から予想されるベクトルを用いることで収束性の改善を図ることが多い [12]．文献 [12] では，薄板弾性体問題 [12], [13] において，SA-AMG 法の適用とその結果が報告されている．文献 [12] における研究では，回転成分 $(0,-z,y)$，$(z,0,-x)$，$(-y,x,0)$（(x,y,z) は節点要素の座標）がニアカーネルベクトルとして用いられている．しかし，このベクトルのみ

Algorithm 3 αSA 法

Given : B_1
Select : $\boldsymbol{x_1}$
for $n = 1$ to *extract_number* **do**
  for $l = 1$ to $L - 1$ **do**
    $\tilde{\boldsymbol{x}}_{\boldsymbol{l}} \leftarrow V\_cycle^{\mu}(A_l \boldsymbol{x_l} = 0)$
    $B_l \leftarrow [B_l, \tilde{\boldsymbol{x}}_{\boldsymbol{l}}]$
    $B_{l+1} \leftarrow P_l^T B_l$
    $\boldsymbol{x_{l+1}} \leftarrow$ Last col. of B_{l+1}
    $Multilevel\_creation(B_{l+1})$
  end for
  for $l = L$ to 2 **do**
    $\boldsymbol{x_{l-1}} \leftarrow P_{l-1} \boldsymbol{x_l}$
  end for
end for
Output $\boldsymbol{x_1}$ as near-kernel vector

extract_number：抽出したい本数
A_{level}：レベル *level* における問題行列 A
$V\_cycle^{\mu}(A\boldsymbol{x} = 0)$：$A\boldsymbol{x} = 0$ を対象に V-cycle を μ 回適用
B_{level}：レベル *level* におけるニアカーネルベクトル候補群の行列
$[B, \boldsymbol{x}]$：行列 B の最終列へのベクトル $\boldsymbol{x}$ の追加
$Multilevel\_creation(B_l)$：行列 B を基に，補間演算子 $P_l, P_{l+1}, \ldots$，および階層行列 $A_l, A_{l+1}, \ldots$ を再生成

で十分なニアカーネルベクトルが設定できているとは限らない．また，これは問題設定に依存したものであり，すべての問題において，対象とする問題の物理的性質に基づき適切なニアカーネルベクトルを予測できるとは限らない．そこで，ニアカーネルベクトルを問題行列から抽出する手法が提案されてきた．この手法により，上記のようなニアカーネルベクトル設定の際に起こる問題を解消することができると考えられる．

ニアカーネルベクトルを抽出する手法に関しての関連研究はいくつか存在する．まず，Brezina らにより提案された αSA 法である [6]．αSA 法は，問題行列から V-cycle（Multigrid 法の解法部）を用いてニアカーネルベクトルを抽出する手法である．αSA 法の概要を Algorithm 3 に示す．Algorithm 3 内の $Multilevel\_creation(B_l)$ は，ニアカーネルベクトル群の行列 B_l を基に，レベル l 以下の粗いレベルにおいて，3.1 節で述べた補間演算子 $P_l, P_{l+1}, \ldots$ の生成，および生成された補間演算子に基づき階層行列 $A_l, A_{l+1}, \ldots$ の再生成を行う処理とする．

αSA 法の概要を図示したものを**図 8** に示す．このように αSA 法は，1 本のベクトルに基づき全階層における行列を網羅するように，ニアカーネルベクトルを抽出する．

そのほかにも，Brezina らにより提案された Spectral AMG 法と呼ばれる手法がある [14]．この手法では，まず有限要素法の単位要素から，その要素ごとにおける小行列を生成する．その後，生成された複数の小行列に対し，それぞれ 0 固有値に近い固有ベクトルを求める．最後に，以上のように各要素で計算されたベクトルを，それぞれの要素番号に対応するように組み合わせ，ニアカーネルベクトルとする手法である．

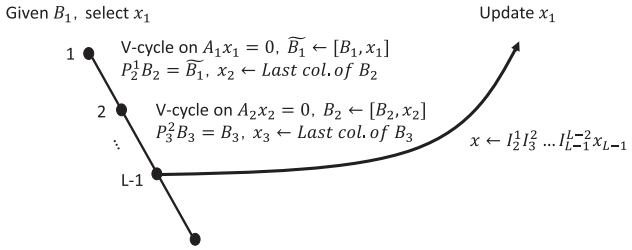

図 8　αSA 法の概要
Fig. 8　The summary of αSA method.

Algorithm 4 著者らの先行研究 [7] におけるニアカーネルベクトル抽出手法

Given : B
Select : $\boldsymbol{x}$
for $n = 1$ to *extract_number* **do**
  $\boldsymbol{x} \leftarrow Random()$
  $\tilde{\boldsymbol{x}} \leftarrow V\_cycle^{\mu}(A\boldsymbol{x} = 0)$
  $B \leftarrow [B, \boldsymbol{x}]$
  $Multilevel\_creation(B)$
end for
Output B matrix as near-kernel vectors

extract_number：抽出したい本数
A：与えられた問題行列 A
$V\_cycle^{\mu}(A\boldsymbol{x} = 0)$：$A\boldsymbol{x} = 0$ を対象に V-cycle を μ 回適用
B：ニアカーネルベクトル候補群の行列
$[B, \tilde{\boldsymbol{x}}]$：行列 B の最終列へのベクトル $\boldsymbol{x}$ の追加
$Multilevel\_creation(B)$：行列 B を基に，補間演算子 $P_1, P_2, \ldots$，および階層行列 $A_1, A_2, \ldots$ を再生成

3.3.2　著者らによる先行研究

著者らは，ニアカーネルベクトルを用いることによる SA-AMG 法の収束性への影響について，与えられた問題行列が置かれるレベル 1 が，最もニアカーネルベクトル設定による影響を与えるのではないかと考えた．また αSA 法では，全階層の行列に対するニアカーネルベクトルを 1 本のベクトルで作成する．しかし，全階層を 1 本のベクトルのみに基づいて，表現することは難しいのではないかと考えた．そこで，抽出時間の削減も含め，ニアカーネルベクトル抽出の効率化を図るため，著者らによる先行研究として，αSA 法を基に，レベル 1 のみにおいてニアカーネルベクトルの抽出を行う手法を提案し，収束性や実行時間に関する計測および評価を行った [7]．Algorithm 4 に文献 [7] における抽出手法の概要を示す．この手法では Algorithm 4 のように，まず初期ベクトル $\boldsymbol{x}$ を乱数で初期化を行い（4 行目），$A\boldsymbol{x} = 0$ を対象に V-cycle を μ 回繰り返す（5 行目）．この μ はユーザ側が与えるパラメータである．その後，ニアカーネルベクトル候補群である行列 B に，V-cycle により出てきたベクトル $\boldsymbol{x}$ を入力する（6 行目）．その後 αSA 法と同様に，抽出されたニアカーネルベクトル群の行列 B を使用して，3.1 節で示した補間演算子 P の生成，および

粗いレベルの行列生成を再度全レベルで行い，階層行列の再生成を行う（7 行目）．これを抽出したい本数分繰り返す（3 行目から 8 行目）．最後に，ニアカーネルベクトル候補群である行列 B を，ニアカーネルベクトルとして出力する．このようにして，レベル 1 の与えられた問題行列 A に対して，ニアカーネルベクトルの複数本抽出を実現している．本手法では最初に行列 B を与えることとなっているが，これはニアカーネルベクトルとして考えられるベクトルを入力する（たとえば，3 次元弾性体問題では平行移動や回転成分を行列 B に与えている）．

$A\boldsymbol{x} = 0$ を V-cycle で近似的に解くことにより，V-cycle で減衰されない収束しにくい成分が残る．これを SA-AMG 法にニアカーネルベクトルとして追加設定し，それらを基に階層行列を再生成し，さらに再度 V-cycle で近似的に解く．これにより，SA-AMG 法における設定したニアカーネルベクトルの成分が効率良く減衰される性質により，設定されたニアカーネルベクトルの成分が除かれた，新たなニアカーネルベクトルが抽出されると期待できる（誤差の影響で，厳密には独立とならない）．これを実現するために，5 行目で抽出されたニアカーネルベクトルを用いて，階層行列の再構成を行っている．本研究ではニアカーネルベクトルが抽出されるたびに，余分な成分を除去するため QR 分解を施し，抽出された各ベクトルが 1 次独立となるような処理を行っている．これにより，本手法を用いることで，適切なニアカーネルベクトルを効率良く複数本抽出できると予想される．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なニアカーネルベクトルを抽出していくこととなる．

上記のようにニアカーネルベクトルを設定する手法はいくつか存在する．本研究では文献 [7] の手法を基に新たなニアカーネルベクトル抽出手法を提案し，計測および有用性の評価を行った．次章より，本研究で用いたニアカーネルベクトル抽出手法について述べる．

4. ニアカーネルベクトル抽出手法の提案

本章では，新たなニアカーネルベクトル抽出手法の提案を行う．まず，提案手法の概要を説明し，数値実験を交え有用性の検証を行う．

4.1 本研究で提案するニアカーネルベクトル抽出手法

この節では，本研究で提案する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．

SA-AMG 法では通常，細かいレベルのニアカーネルベクトルを基に，粗いレベルのニアカーネルベクトルを生成する．そのため，文献 [6] や [7] の手法では，最終的にレベル 1 としてのニアカーネルベクトルのみを出力しているため，全階層でニアカーネルベクトルの本数が同じとなる．しかし著者らは，粗いレベルのニアカーネルベクトルは，細かいレベルのニアカーネルベクトルだけでは不十分であり，粗いレベルにおいても抽出や追加に設定を行えるようにするべきではないかと考えた．図 4 で示したように，粗いレベルのニアカーネルベクトルは細かいレベルのニアカーネルベクトルを基に生成する．しかし，粗いレベルの行列は細かいレベルの行列とは異なるため，粗いレベルのニアカーネルベクトルとして適切であるかは不明である．また，そもそも細かいレベルにおけるニアカーネルベクトルの設定が不十分である可能性がある．この場合，粗いレベルにおいてニアカーネルベクトルを個別に追加設定することで，計算コストの増大を低く抑えつつ，細かいレベルで行えなかったニアカーネルベクトル成分の減衰を，効率良く行うことができると期待できる．そこで本研究では文献 [7] の手法を基に，粗いレベルにおいてもニアカーネルベクトルを抽出および SA-AMG 法で追加的に設定する手法の提案，および収束性や実行時間の評価を行った．Algorithm 5 に具体的な流れを示す．Algorithm 5 の赤字箇所は，Algorithm 4 から追加された箇所である．Algorithm 5 から分かるように，本手法は Algorithm 4 を基に，粗いレベルにおいてもニアカーネルベクトルを複数本抽出できるように改良を加えた手法となっている．本手法ではニアカーネルベクトル候補群である行列 B を各階層で用意しており，最終的に各階層ごとのニアカーネルベクトルがそれぞれ出力される．これらのニアカーネルベクトルを SA-AMG 法に用いる際には，まずは先行研究による手法と同様に，最も細かいレベルにおいて抽出されたニアカーネルベクトルを設定し，次の粗いレベルの行列とニアカーネルベクトルを生成する．そして，粗いレベルでも

Algorithm 5 本研究でのニアカーネルベクトル抽出提案手法

Given : B_1
Select : x_1
for $level = 1$ to $max\_level - 1$ **do**
  for $n = 1$ to $extract\_number$ **do**
    $\tilde{\boldsymbol{x}}_{\boldsymbol{level}} \leftarrow Random()$
    $\boldsymbol{x}_{\boldsymbol{level}} \leftarrow V\_cycle^{\mu}(A_{level}\tilde{\boldsymbol{x}}_{\boldsymbol{level}} = 0)$
    $B_{level} \leftarrow [B_{level}, \boldsymbol{x}_{\boldsymbol{level}}]$
    $Multilevel\_creation(B_{level})$
  end for
end for
Output $B_1, B_2, \ldots$ matrices as near-kernel vectors

$Random()$：乱数生成
max_level：階層の最大レベル数
$extract\_number$：抽出したい本数
A_{level}：レベル $level$ における問題行列 A
$V\_cycle^{\mu}(A\boldsymbol{x} = 0)$：$A\boldsymbol{x} = 0$ を対象に V-cycle を μ 回適用
B_{level}：レベル $level$ におけるニアカーネルベクトル候補群の行列
$[B, \boldsymbol{x}]$：行列 B の最終列へのベクトル $\boldsymbol{x}$ の追加
$Multilevel\_creation(B_l)$：行列 B_l を基に，補間演算子 $P_l, P_{l+1}, \ldots$，および階層行列 $A_l, A_{l+1}, \ldots$ を再生成

表 4　各論文における手法の比較と実験環境設定
Table 4　Comparing of each method.

	[Brezina et al., 2004] [6]（αSA）	[Nomura et al., 2016] [7]	本研究
各レベルの設定本数	一定	一定	変更
抽出方法	全階層を 1 本のベクトルに基づき表現	レベル 1 のみ抽出	各階層で抽出
設定本数	最大 6	最大 10	最大 10（レベル 1） 最大 15（レベル 2） 最大 20（レベル 3）
プロセス数	4	512	512
問題サイズ	200,000	5,120,000	5,120,000

同様に，細かいレベルのニアカーネルベクトルを基に，さらに粗いレベルを生成する．本手法ではこの際，細かいレベルをもとに生成されたニアカーネルベクトルに追加して，抽出されたニアカーネルベクトルを用いる．この操作を最大レベル数 − 1 まで行う．以上のように本手法では，文献 [7] で用いた手法に加え粗いレベルのニアカーネルベクトルも考慮しているため，粗いレベルの行列が全体の収束性に大きな影響を与えている場合，文献 [7] よりもさらに収束性が改善することが見込める．

外部から入力するパラメータとして，*extract_number* と μ が存在する．*extract_number* は抽出したい本数であり，著者らによる研究 [7] から，適切な本数を設定することが必要であることが分かっている．この値に関しては，5.1 節で，予測する手法の検討を行う．また μ は，V-cycle を繰り返す回数であり，この値により収束性が変化すると考えられるが，本事項については今後の課題とし，本研究では μ を 20 に設定し，計測を行っている．

ここで，ここまでの手法に関してまとめた表を**表 4** に示す．本研究での提案手法であることを強調するために，表 4 の本研究の箇所を赤字で示している．表 4 から分かるように，本研究の手法における最も大きな点としては，各階層でニアカーネルベクトルを抽出，および設定本数の変更が可能となったことである．次節の数値実験において，粗いレベルにおけるニアカーネルベクトル抽出本数の変更による，収束性や実行時間への影響についても示す．

4.2　数値実験

本節では，提案手法を用いることによる数値実験の結果を示す．本節ではまず，本実験で用いた環境について述べ，その後，数値実験の内容とその結果を示す．

4.2.1　実験環境

本研究では，東京大学情報基盤センターと筑波大学計算科学研究センターが共同運営する，最先端共同 HPC 基盤施設（JCAHPC：Joint Center for Advanced High Performance Computing）による，Oakforest-PACS スーパーコンピュータシステムを使用し数値実験を行った．Oakforest-PACS は，1 ノードに 1 個の Intel® Xeon Phi™ プロセッサ（68cores，1.4 GHz）と，16 GB の MCDRAM と 96 GB の DDR4 メモリを搭載している（本研究では，MCDRAM のみ用いる設定を行い，数値実験を行った）．また，Oakforest-PACS では Intel® Omni-Path（12.5 GB/s）により，各計算ノードを接続している．

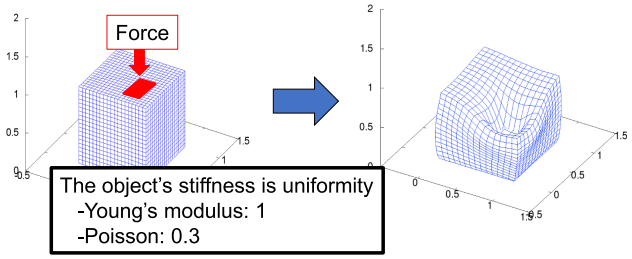

図 9　数値実験で使用した 3 次元弾性体の問題設定
Fig. 9　Setup of the experimental problem in elasticity at numerical experiment.

数値実験では最大 8 ノード使用し，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の，物体の変形量を解く問題となっている．本研究では，単純な立方体メッシュによる有限要素法を用いて離散化を行った．またこの問題は 3 次元なため，係数行列の生成時には 1 節点あたりの行列要素が 3×3 のブロック行列に表現される．アグリゲートの生成は節点単位で行うため，3×3 のブロック行列においてフロベニウスノルムを計算してから，節点集合の作成を行っている．数値実験で用いた弾性体の問題設定を**図 9** に示す．この弾性体の問題は図 9 のように，均質な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，すべての部分で 1 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．また，ポアソン比（物体のひずみに関する係数）を 0.3 に設定した．問題サイズについては，1 プロセスあたり $15 \times 15 \times 15$ のウィークスケーリング（Weak Scaling）方式による計測を行った．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．

本研究では，AMGS ライブラリ [15] を使用して実験を行っている．AMGS ライブラリは，著書らが研究開発を行っている，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 CG 法 [16] を使用し，前処理として SA-AMG 法を適用している．SA-AMG 法における解法部の緩和法には，対称 Gauss-Seidel 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，反復回数の上限を 500 回とした．AMGS ライブラリと解法部の CG 法は，Fortran を用い作成を行っている．また，コンパイラは Intel® コンパイラ（mpiifort），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 “-O3” と “-xMIC-AVX512”，OpenMP を用いるためのオプションである “-qopenmp” を使用した．

4.2.2　数値実験内容

本実験における比較対象を**表 5** に示す．本実験では，粗いレベルで抽出を行うことによる効果の検証のため，著者らの研究で従来用いていた粗いレベルで抽出を行わない手法（レベル 1 のみ）と，今回の手法である粗いレベルで抽出を行う手法（粗いレベルで抽出）で比較を行っている．また，本実験では第 1 レベルのニアカーネルベクトルに関しては，**表 6** に示すような本数の抽出および設定をそれぞれ行った．表 6 の 3p と 6p は弾性体問題の問題設定から予想されるニアカーネルベクトルである．また，第 1 レベルで抽出したニアカーネルベクトルを用いる際には，3p に追加する形で設定を行っている（3p+1，3p+2，...）．ここで，本実験のニアカーネルベクトル設定に関する具体例を示した図を**図 10** に示す．図 10 は本研究の手法において，第 1 レベルでニアカーネルベクトルを 4 本（3p+1）設定した際に，「粗いレベルで抽出」を適用した際の各レベルのニアカーネルベクトル設定本数を示した例となっている．この図のように，「レベル 1 のみ」では，細かいレベルを基に次のニアカーネルベクトルを生成するため，すべてのレベルで同じ本数となる．一方，「粗いレベルで抽出」では，「+1」の場合はレベル数が増加するごとに，ニアカーネルベクトルの設定本数を 1 本ずつ増やしていく．「+5」では，5 本ずつとなる．

表 5　粗いレベルにおけるニアカーネルベクトル抽出手法の実験における比較対象

Table 5　The target of comparison in preliminary experiment.

比較対象	詳細
レベル 1 のみ	粗いレベルで抽出なし [7]
粗いレベルで抽出：+1，+5	Level2 以降の階層ごとに 1（+1）または 5（+5）本抽出，追加設定

表 6　第 1 レベルのニアカーネルベクトル抽出本数

Table 6　The settings of near-kernel vectors at level 1.

表記	詳細
3p	平行移動成分 X，Y，Z（3 本）
6p	3p（3 本）+回転成分 X，Y，Z（3 本）
3p+1，3p+2，...	3p（3 本）+第 1 レベルの抽出本数（最大 7 本）

4.2.3　実験結果

本項では，本研究の提案手法である，粗いレベルにおけるニアカーネルベクトル抽出手法（Algorithm 5）の実験結果を示す．数値実験による結果を**図 11** と**図 12** に示す．図 11 と図 12 はそれぞれ 1 並列，512 並列時の，ニアカーネルベクトルの設定本数による反復回数および実行時間の変化を示している．これらのグラフでは，比較対象として 3p と 6p を設定した際の結果も記載している．まず反復回数に着目すると，本研究の提案手法（粗いレベルで抽出）を用いることで，反復回数の改善がみられることが分かる．これは，粗いレベルで追加的にニアカーネルベクトルを設定することで，粗いレベルの行列においても効率的に誤差成分の減衰が行え，全体の反復回数が削減されたものと考えられる．次に全体の実行時間に着目しても，1 並列においては提案手法を用いることで，実行時間が改善されることが分かる（「レベル 1 のみ」の最良値である 3p+3 の場合と，粗いレベルで抽出の最良値である「粗いレベルで抽出：+5」の 3p+3 を比較すると，「粗いレベルで抽出」を用いることで約 10%の削減効果）．しかし，1 並列における 3p+4 以降，および 512 並列では，6p の実行時間と比べ悪化していることが分かる．これは，ニアカーネルベクトル本数の増加にともない，構築部における粗いレベルの行列生成にかかる時間が増加し，結果として収束性改善の効果以上に計算時間が悪化したためであると考えられる．実際，破線で示した解法部のみの実行時間を見ると，「粗いレベルで抽出」が「レベル 1 のみ」よりも良い結果が得られ

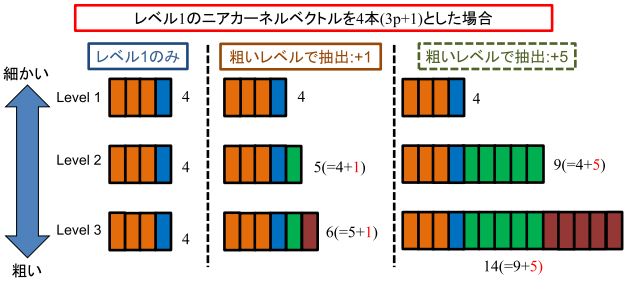

図 10　「粗いレベルで抽出」適用時の SA-AMG 法におけるレベルごとのニアカーネルベクトル設定

Fig. 10　The way how to set near-kernel vectors at each level in “the extraction at coarser levels”.

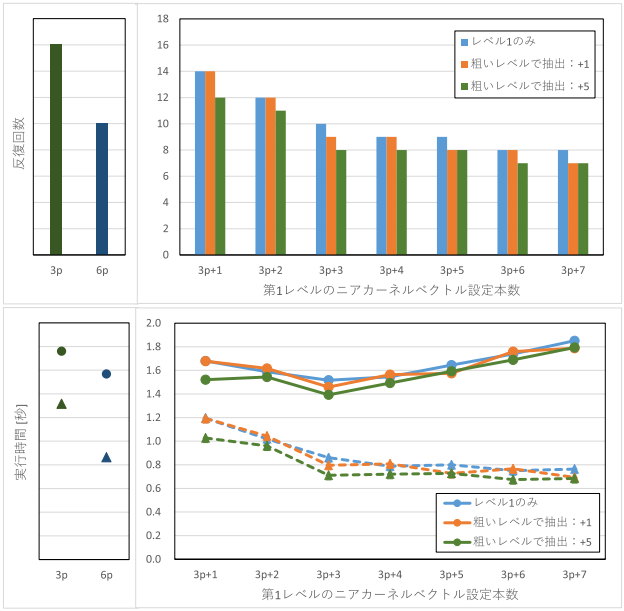

図 11　ニアカーネルベクトルの設定本数による反復回数（上）と実行時間（下）の変化：1 並列（下図における実線（丸マーカ）は全体（構築部 + 解法部），破線（三角マーカ）は解法部のみ）

Fig. 11　The number of iteration (upper) and the execution time (lower) by changing the number of near-kernel vectors: 1 process (In lower graph, the solid lines (round marker) show the whole execution time (setup + solution part), and the dashed lines (triangle marker) show the execution time of only solution part).

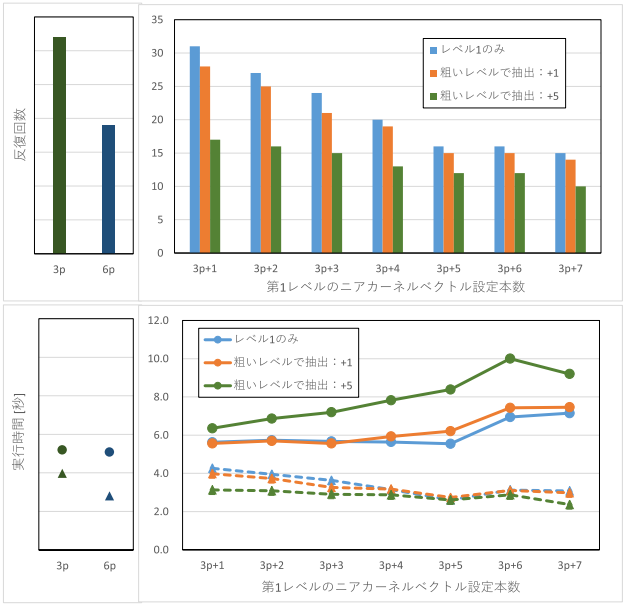

図 12　ニアカーネルベクトルの設定本数による反復回数（上）と実行時間（下）の変化：512 並列（グラフ要素は図 11 と同様）

Fig. 12　The number of iteration (upper) and the execution time (lower) by changing the number of near-kernel vectors: 512 process (The details is the same as Fig. 11).

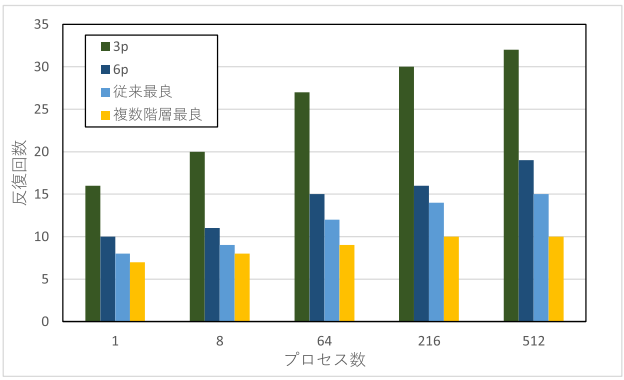

図 13　プロセス数変化による反復回数（「従来最良」と「複数階層最良」は，それぞれ「レベル 1 のみ」と「粗いレベルで抽出」における反復回数に着目したときの，最良本数設定時の結果）

Fig. 13　The number of iteration by changing the number of processes (“The best of previous” and “The best of multilevel” are the result when the optimum number of near-kernel vector is set at “Only level 1” and “Extraction at coarser levels”, respectively).

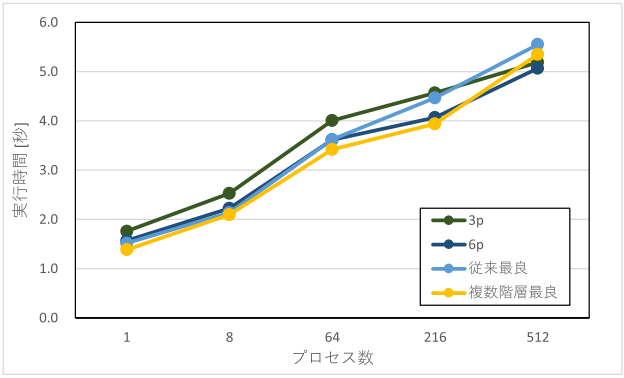

図 14　プロセス数変化による実行時間（グラフ要素は図 13 と同様）

Fig. 14　The execution time by changing the number of processes (The details is the same as Fig. 13).

ることが分かる．このように，抽出したニアカーネルベクトルを追加で設定することによる，収束性改善と計算コスト増加のトレードオフが発生することが分かる．

ここで，プロセス数を変化させることによる反復回数および実行時間を**図 13** と**図 14**，および**図 15** に示す．グラフ内の要素である「従来最良」は，「レベル 1 のみ」において，「複数階層最良」は「粗いレベルで抽出」において最良のものを設定した際の値となっている（例として図 14 の 1 プロセスにおいて図 11 と対応させると，「従来最良」は「レベル 1 のみ」の 3p+3 の値を示し，「複数階層最良」では「粗いレベルで抽出：+5」の 3p+3 の値を示している）．さらに「従来最良」および「複数階層最良」について，図 13 は反復回数，図 14 は全体の実行時間，図 15 は解法部のみの実行時間にそれぞれ着目したときに，最良となる設定を行った場合の結果である（そのため，図 13，図 14，図 15 でニアカーネルベクトル設定本数がそれぞれ異なる）．図 13 より，粗いレベルでニアカーネルベクトル

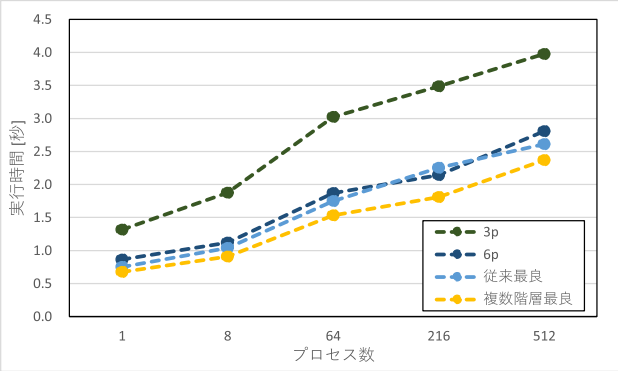

図 15　プロセス数変化による解法部のみの実行時間（グラフ要素は図 13 と同様）

Fig. 15　The execution time of only solution part by changing the number of processes (The details is the same as Fig. 13).

を抽出することにより，従来手法と比べ収束性の改善がみられることが分かる（複数階層最良を用いることで，512 並列において「6p」と比べ約半分，「従来最良」と比べても約 30%の削減）．これより，粗いレベルでニアカーネルベクトルを適切な本数抽出および設定を行うことで，高並列・大規模問題においてもニアカーネルベクトル成分を効率良く減衰させることができ，結果として収束性の改善がみられることが分かった．次に図 14 に着目すると，216 並列までは複数階層最良が最も良い結果が得られていたが，512 並列時では 6p と比べ悪化していることが分かる．しかし，図 15 に着目すると，複数階層最良が全体的に最も良い結果を示すことも分かる．これは図 12 でも述べたように，ニアカーネルベクトルの設定本数を増やすことによる構築部の計算コスト増大によるものである．さらに，図 14 および図 15 において用いたニアカーネルベクトルの本数構成を，**表 7** と**表 8** にそれぞれ示す．表 7 と表 8 における複数階層最良において，「3p+3：+4」のように表記されているが，これはレベル 1 において 3 本抽出，粗いレベルにおいて 4 本ずつ抽出したことを示す．表 7 より，ニアカーネルベクトルを少数設定することで，構築部における計算コストが抑えられ，全体の実行時間が改善される傾向があることが分かる．また，表 8 より，ニアカーネルベクトルを多数設定することで収束性が改善し，結果として解法部の実行時間の改善につながることが分かる．

上記より，ニアカーネルベクトルを適切な本数設定することで，Algorithm 5 の手法は有用となることが分かる．しかし，適切な本数の検証には，Algorithm 5 の手法ではすべてのニアカーネルベクトルのパターンを試す必要がある（たとえば図 11 の場合，1 レベル目のニアカーネルベクトルを 7 本，2 レベル目以降は 1 本と 5 本，および抽出なしの 3 通り存在し，本実験においても合計 15 通り存在する．ニアカーネルベクトルのとりうる本数は最大で行列サイズ分あり，すべてのパターンを網羅することは不可能で

表 7　図 14 におけるニアカーネルベクトル本数の構成

Table 7　The construction of the number of near-kernel vectors at Fig. 14

	1	8	64
従来最良	3p+3	3p+3	3p+3
複数階層最良	3p+3：+4	3p+3：+3	3p+1：+3
	216	512	—
従来最良	3p+1	3p+5	—
複数階層最良	3p+1：+3	3p+1：+2	—

表 8　図 15 におけるニアカーネルベクトル本数の構成

Table 8　The construction of the number of near-kernel vectors at Fig. 15.

	1	8	64
従来最良	3p+6	3p+5	3p+5
複数階層最良	3p+6：+5	3p+6：+5	3p+7：+5
	216	512	—
従来最良	3p+4	3p+5	—
複数階層最良	3p+6：+5	3p+7：+5	—

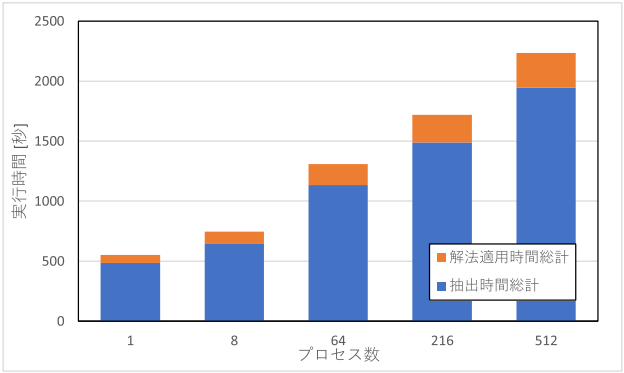

図 16　適切なニアカーネルベクトル設定本数検証にかかる時間

Fig. 16　The execution time to verify optimum number setting of near-kernel vectors.

ある）．実際に，図 13 と図 14，および図 15 における，「複数階層最良」の設定本数の検証にかかった時間を**図 16** に示す．抽出適用時間総計の割合が解法適用時間総計に比べ大きいが，これは主に 2 つ要因がある．1 つは，ニアカーネルベクトルの本数増加にともない，階層行列の再構成（Algorithm 5 における $Multilevel\_creation(B_{level})$）のコストが増大してしまうためである．もう 1 つは，細かいレベルのニアカーネルベクトルが変更された場合，粗いレベルの行列がすべて変化してしまうため，そのつど粗いレベルの行列に対して，再抽出を行う必要があるためである．このグラフより，検証にかかる時間は実際の実行時間と比べ約 400〜500 倍となることが分かる．現状では，本手法を実際の問題に適用する場合に，検証に膨大な時間が必要となり，実用上の観点から問題がある．次章ではこの問題の解決に向け，適切なニアカーネルベクトルの本数を予測する方法を組み込んだ手法をさらに提案し，数値実験によ

る検証を行う．

5. 適切なニアカーネルベクトル設定本数予測手法付き抽出手法の提案

5.1 適切なニアカーネルベクトル設定本数の予測

本節ではさらに，ニアカーネルベクトル設定本数（Algorithm 4 や Algorithm 5 における $extract\_number$）の適切な設定本数を予測する手法を検討する．前章の実験結果や，著者らによる先行研究 [7] により，Algorithm 4 や Algorithm 5 の手法において，ニアカーネルベクトルを適切に設定することで，収束性や実行時間において有用となる．しかし，適切なニアカーネルベクトル本数の検証には，抽出本数のとりうるすべてのパターンを網羅する必要がある．4.2.3 項より，現状では検証に膨大な時間を必要としてしまい，実用面からみて大きな問題となる．そこで本研究では，文献 [6] で用いられていた判定基準値を Algorithm 5 の手法に組み込み，ニアカーネルベクトルの抽出を行った．具体的な計算式は以下のとおりとなる．

$$\{(A_{level}\boldsymbol{x}_{level}, \boldsymbol{x}_{level})/(A_{level}\hat{\boldsymbol{x}}_{level}, \hat{\boldsymbol{x}}_{level})\}^{1/\mu} \qquad (4)$$

ここで，$(\cdot,\cdot)$ の形で表されている式は，ベクトルどうしの内積を示す．上記の式において，ニアカーネルベクトルを n 本抽出した場合，$\boldsymbol{x}$ は n 本目，$\hat{\boldsymbol{x}}$ は $n-1$ 本目に抽出したベクトルを示す．また，$level$ は level 番目の階層における行列やベクトルを示す．SA-AMG 法は 3.1 節で説明したように，ニアカーネルベクトルを設定することで，残差が効率良く減衰する．この性質を利用し，十分なニアカーネルベクトルが抽出された，つまり抽出時における $A\boldsymbol{x}=0$ の求解による残差が早く減衰した場合，ニアカーネルベクトルの抽出を終了する．上記の判定基準値において，n 本目に抽出したニアカーネルベクトル $\boldsymbol{x}$ による行列ベクトル積 $A\boldsymbol{x}$ が，$n-1$ 本目の $\hat{\boldsymbol{x}}$ による行列ベクトル積 $A\hat{\boldsymbol{x}}$ よりも十分 0 に近くなれば，十分にニアカーネルベクトルが抽出されたと見なすことができ，その場合式の結果が 0 に近くなる．つまり，判定基準値の結果が十分小さい値を示した際に，これ以上のニアカーネルベクトルの追加による，収束性改善の効果が見込めないと判断できる．この際ニアカーネルベクトルの抽出を終了することで，過剰なニアカーネルベクトルの抽出を抑えられると期待できる．

Algorithm 6 に，予測手法を用いたニアカーネルベクトル抽出手法の概要を示す．Algorithm 6 の青字で示した箇所は Algorithm 5 からの追加箇所である．Algorithm 6 において新たな値 ϵ がある．これは，ユーザ側が与えるパラメータとなっており，この値よりも判定基準値が小さい場合，そのレベルにおける抽出は十分であると判断し，次のレベルの抽出に移る．次節で，Algorithm 6 の手法による有用性を数値実験により示す．

Algorithm 6 予測手法付きニアカーネルベクトル抽出手法

```
Given : B_1
Select : x_1
for level = 1 to max_level − 1 do
  for n = 1 to extract_number do
    x̃_level ← Random()
    x_level ← V_cycle^μ(A_level x̃_level = 0)
    if (A_level x_level, x_level)/
    (A_level x̂_level, x̂_level)}^{1/μ} < ε_level then break
    B_level ← [B_level, x_level]
    Multilevel_creation(B_level)
  end for
end for
Output B_1, B_2, ... matrix as near-kernel vectors
```

$Random()$：乱数生成
max_level：階層の最大レベル数
$extract\_number$：抽出したい本数
A_{level}：レベル $level$ における問題行列 A
$V\_cycle^{\mu}(A\boldsymbol{x}=0)$：$A\boldsymbol{x}=0$ を対象に V-cycle を μ 回適用
B_{level}：レベル $level$ におけるニアカーネルベクトル候補群の行列
$[B, \boldsymbol{x}]$：行列 B の最終列へのベクトル $\boldsymbol{x}$ の追加
$Multilevel\_creation(B_l)$：行列 B を基に，補間演算子 $P_l, P_{l+1}, \ldots$，および階層行列 $A_l, A_{l+1}, \ldots$ を再生成
ϵ_{level}：判定基準値の閾値

5.2 数値実験

本節では，予測手法付きニアカーネルベクトル抽出手法における数値実験の結果を示す．実験環境は 4 章と同様であるため，割愛する．以下より，数値実験の内容とその結果を示す．

5.2.1 数値実験内容

本実験では，Algorithm 6 の手法の有用性検証に向けた数値実験の内容について述べる．この実験ではまず予備実験として，反復回数と予測式の関係を，問題サイズ $10\times10\times10$ の小規模な問題上において示す．次に，ウィークスケーリング方式による実験で，Algorithm 6 を用いることによる有用性を示す．ここで Algorithm 6 および 5.1 節で述べたように，本手法では閾値 ϵ を設定する必要がある．本研究では閾値設定の際，各レベルで異なる値を設定し，計測を行った．これについての具体的な設定数値を**表 9** および**表 10** に示す．表 9 および表 10 のように，本研究では階層行列の最大レベル数に応じて，閾値 ϵ の設定を変更している（本実験ではウィークスケーリング方式による計測であるため，プロセス数の増加とともにレベル数が増加する．今回の場合，1 並列および 8 並列では 3 レベル，64 並列および 216 並列では 4 レベル，512 並列は 5 レベルとなる）．さらに，レベル数が下がるごとに，閾値 ϵ が低くなるように設定を行っている．著者らによる研究から，細かいレベルは問題サイズが大きいため，ニアカーネルベクトル設定本数の増加により，計算コストが大きく増加し，結果として全体実行時間へ大きく影響してしまうことも分

表 9　各レベルにおける ϵ の値（全体実行時間優先）
Table 9　The value of ϵ at each level.

最大レベル数	ϵ の値			
	Level 1	Level 2	Level 3	Level 4
3	0.250	0.156	—	—
4	0.400	0.250	0.156	—
5	0.640	0.400	0.250	0.156

表 10　各レベルにおける ϵ の値（収束性優先）
Table 10　The value of ϵ at each level.

最大レベル数	ϵ の値			
	Level 1	Level 2	Level 3	Level 4
3	0.100	0.070	—	—
4	0.143	0.100	0.070	—
5	0.204	0.143	0.100	0.070

かっている．これを緩和するため表 9 および表 10 のように，本実験では粗いレベルにおいて，閾値 ϵ を低く設定することでニアカーネルベクトル抽出本数を増やし，ニアカーネルベクトル成分の減衰をより厳密に行うようにした．また，閾値 ϵ の値を高く設定することで，ニアカーネルベクトルの抽出本数が少なくなる．これにより，計算コストが減少するが，収束性が悪化することが考えられる．これを検証するために，表 9 は閾値 ϵ を低く設定，表 10 は高く設定しており，それぞれの閾値 ϵ による検証実験も行った．ここまで閾値 ϵ について述べてきたが，表 9 および表 10 のような設定は，以上のような経験則に基づいており，すべての問題で同じ設定が適用できるかは不明である．そのため，閾値 ϵ の設定手法の確立は今後の課題とする．

5.2.2　実験結果

本項では，予測手法付き抽出法（Algorithm 6）における結果を示す．まず，予備実験における結果を**図 17** に示す．「予測式」については，3p+n 本目を抽出した際に計算されたときの予測式の結果を示している（例として 3p+1 における値は，1 本目を抽出した際に算出された値となっており，本実験では約 0.22 を示した）．図 17 より，3p+3 から 3p+4 にかけて，反復回数の改善が効率良く行われていないことが分かる．これより，ニアカーネルベクトルの設定本数増加により計算コストが増大することを考慮すると，本実験の問題設定では 3p+3 が適切な本数設定である．それと同時に，3p+3 から 3p+4 にかけて予測式の結果が大きく下がっていることが分かる．これらより，本実験の問題設定では，閾値 ϵ を 0.1 に設定することで，予測式による適切な本数の設定が可能となる．これは実際にすべてのパターンを計測したため判明した事実であり，5.2.1 項でも述べたように，閾値 ϵ の設定は今後の課題とし，次の実験では表 9 および表 10 の値を使用した．

次の実験では，実際に Algorithm 6 の手法を用いたウィークスケーリングにおける結果を示す．この実験による実

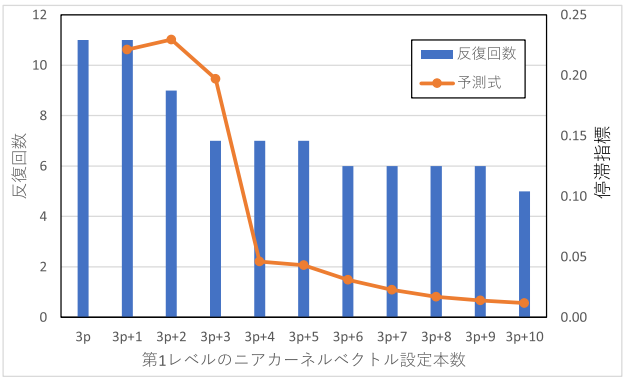

図 17　予備実験：予測式と反復回数の結果（問題サイズ：$10 \times 10 \times 10$，1 並列）
Fig. 17　Preliminary experiment: the result of stagnation indicator and the number of iterations (Problem size: $10 \times 10 \times 10$, single process).

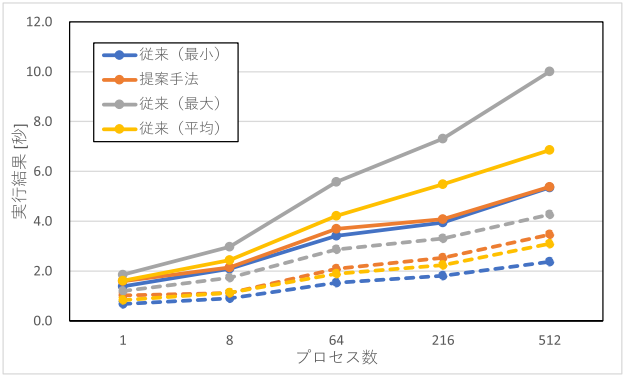

図 18　Algorithm 6 の手法を用いた際の実行時間の結果（表 9 の閾値 ϵ を使用．実線は全体の実行時間，構築部 + 解法部，破線は解法部のみの時間を示す）
Fig. 18　The execution time by using Algorithm 6 (Using ϵ of Table 9. The solid lines (round marker) show the whole execution time (setup + solution part), and the dashed lines (triangle marker) show the execution time of only solution part).

行時間を**図 18** および**図 19** に示す．グラフの要素である「従来」は，図 11 や図 12 のように，1 レベル目を 7 本，2 レベル目以降を 5 本抽出するまでのすべて組合せを試し，実行時間においてすべての組合せにおける最小，最大，平均を算出したものとなる（つまり，（最小）は提案手法における理想の結果であり，提案手法を用いることで低コストで（最小）に近づくことが目的となる）．図 18 より，全体の実行時間を考え，構築部の実行時間を抑えるために閾値を高く設定した場合，全体の実行時間は従来（最小）とほぼ同等となることが分かる．しかし解法部の実行時間に着目すると，提案手法は従来（最小）と比べ悪化していることが分かる．また図 19 より，解法部の実行時間を考え，逆に閾値を低く設定した場合，全体の実行時間は従来手法と比べ悪化しているが，解法部の実行時間は従来（最小）と同等か，さらに改善されることが分かる．これらの結果

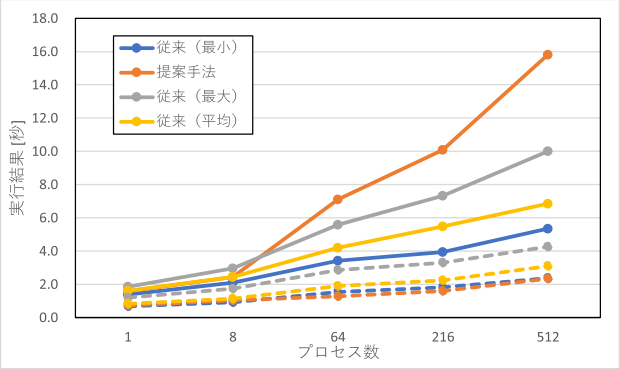

図 19　Algorithm 6 の手法を用いた際の実行時間の結果（表 10 の閾値 ϵ を使用．グラフの要素は図 18 と同様）

Fig. 19　The execution time by using Algorithm 6 (Using ϵ of Table 10. The details is the same as Fig. 18).

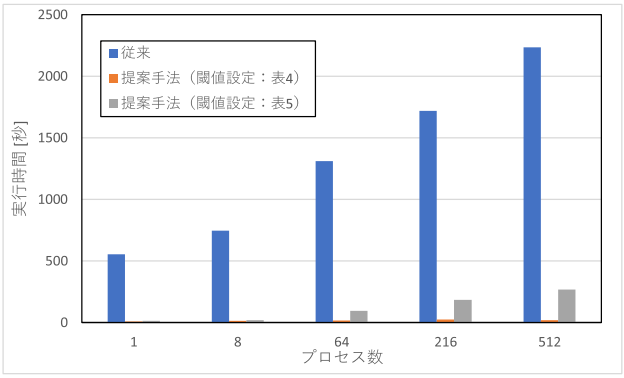

図 20　Algorithm 6 の手法による適切なニアカーネルベクトル設定本数検証にかかる時間

Fig. 20　The extraction time to verify optimum number setting of near-kernel vectors by using Algorithm 6.

表 11　Algorithm 6 の手法による反復回数

Table 11　The number of iteration by Algorithm 6.

	1	8	64	216	512
従来（最小）	7	8	9	10	10
提案手法（閾値 ϵ 設定：表 9）	12	11	17	20	24
提案手法（閾値 ϵ 設定：表 10）	8	10	7	7	8

から，閾値を高く設定した場合，ニアカーネルベクトルの抽出本数を抑えるため，構築部の実行時間が改善するが，収束性が悪化し解法部の実行時間が増大する．逆に，低く設定するとその逆の効果が得られることが分かる．実際，**表 11** からもその事実は明らかである．表 11 は提案手法をそれぞれの閾値設定で実行した場合の反復回数を示している．この表より，上記で述べたように，閾値を高く設定した表 9 では反復回数が多く，閾値を低く設定した表 10 は反復回数が少ないことが分かる．ここで，提案手法（閾値 ϵ 設定：表 10）における結果が従来（最小）よりも改善しているが，これは従来の結果では 1 レベル目を 7 本，2 レベル目以降を 5 本抽出するまでの組合せまでのみしか検証していないが，提案手法により，より多くのニアカーネルベクトルを使用するという判断をしたためである．このように提案手法を用いることで，パターンが膨大であり通常では検証できない組合せにおいても，低コストで判断できることが分かる．ここで，前実験と同様に，適切なニアカーネルベクトルの設定本数検証の時間を**図 20** に示す．図 20 より，提案手法を用いることで，従来問題となっていたニアカーネルベクトルの本数設定に対する検証時間が大きく削減されていることが分かる．これより提案手法を用いることで，実際に運用する際にも低コストで容易に運用可能となると考えられる．

以上より，本論文の提案手法では，用途に応じて閾値 ϵ を適切に設定することで，従来手法と比べ低コストで改善がみられることが分かった．より詳細な内容としては，与えられた問題行列が解法適用時に毎回異なる場合には，表 9 のように閾値 ϵ を高く設定する，対して反復回数を多く必要とする悪条件の問題や，元の問題設定は変えずに右辺ベクトル $\boldsymbol{b}$ の設定を変更する（この場合，構築部での粗いレベルの問題作成を 1 回行えば，そのまますべての計算に同じ階層行列を使用できる）場合には，表 10 のように低く設定することで，本手法が高い効果を発揮できると考えられる．

6.　結論

本研究では SA-AMG 法の適用性の拡張に向け，粗いレベルでニアカーネルベクトルを複数本抽出し，抽出時において適切な設定本数を予測する手法の提案を行った．そして，SA-AMG 法において本手法を用いることによる有用性を，従来手法と比較することで検証を行った．本研究では 2 つの実験を行った．まず，ニアカーネルベクトルの設定本数の変化による収束性や実行時間の変化を示した．この実験より，ニアカーネルベクトルを複数本設定することで，反復回数が改善することが分かった．また実行時間に関しても，ニアカーネルベクトルを複数本設定することで，反復回数の改善の影響で解法部の実行時間が改善することも分かった．しかし，階層行列を作成する構築部においては，ニアカーネルベクトルの本数を増やすことで，階層行列生成の計算コストが増加してしまい，結果として構築部と解法部の実行時間の合計が悪化してしまうこともみられた．これらより，ニアカーネルベクトルを複数本設定する際には，適切な本数を設定することが必要であることが分かった．適切な本数の設定には，この手法ではニアカーネルベクトルの本数の組合せを全パターン網羅する必要があるが，この検証には膨大な時間がかかってしまうという問題があった．そこで次の実験では本研究の目的である，粗いレベルでニアカーネルベクトルを複数本抽出し，抽出時において適切な設定本数を予測する手法の提案を行った．この手法では，パラメータとして閾値 ϵ を設定する必要が

ある．この実験より，この値を適切に高く設定することで，構築部と解法部の合計時間を前実験の最良実行時間とほぼ同等に，またこの値を低く設定することで，解法部のみであるが実行時間をほぼ同等かさらに良い結果を示した．また検証時間についても，従来膨大な時間を必要としていたが，本手法によりコストを大きく抑えることに成功した．以上より，本論文の提案手法では，与えられた問題行列が解法適用時に毎回異なる場合には，閾値 ϵ を高く設定する．対して，反復回数を多く必要とする悪条件の問題や，元の問題設定は変えずに右辺ベクトルが毎回異なる場合には低く設定するように，目的に応じて適切なパラメータを設定することで，本手法が高い効果を発揮できると考えられる．

今後の課題として，まずニアカーネルベクトルの本数予測で用いている閾値 ϵ の設定方法の確立，または新たな予測手法の提案が必要であると考えられる．本研究では閾値 ϵ は利用者側が決定することとなっているが，この値により提案手法を用いることによる収束性や実行時間が大きく影響すると考えられる．そのため，予測手法に用いている閾値 ϵ の設定方法の確立，あるいは閾値 ϵ を必要としない新たな予測手法の確立が必要となる．さらに，本研究では3次元弾性体問題のみしか扱っていないが，他の問題に対しても適用し，有用となるか検証していくことも必要であると考えている．本研究の最終目標は SA-AMG 法の適用性の拡張であり，実際に提案手法が他の問題に対しても有用となるかを検証することは，重要な課題である．これについて，著者らはまず異方性のある問題を対象とすることを考えている．異方性のある問題は悪条件であり，反復回数が大きく増加してしまう傾向がある．そのような問題に対して有用性を示すことで，他の問題に対しても有用となることが期待できる．

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*, Vol.179, pp.344–351 (2006).

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

[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] Vanek, P., Brezina, M. and Mandel, J.: Convergence of Algebraic Multigrid Based on Smoothed Aggregation, *Numerische Mathematik 2001*, Vol.88, pp.559–579 (2001).

[5] Fujii, A., Nishida, A. and Oyanagi, Y.: Evaluation of parallel aggregate creation orders: Smoothed aggregation algebraic multigrid method, *High Performance Computational Science and Engineering*, pp.99–122, Springer (2005).

[6] 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).

[7] Nomura, N., Fujii, A., Tanaka, T., Nakajima, K. and Marques, O.: Performance Analysis of SA-AMG Method by Setting Extracted Near-kernel Vectors, *VECPAR2016* (2016).

[8] Stuben, K.: Algebraic multigrid (AMG): An introduction with applications, *Multigrid*, pp.413–532, Academic Press (2000).

[9] Brandt, A.: Algebraic multigrid theory: The symmetric case, *Applied Mathematics and Computation*, Vol.19, No.1-4, pp.23–56 (1986).

[10] Briggs, W.L., Henson, V.E. and McCormick, S.F.: *A multigrid tutorial*, SIAM (2000).

[11] 杉原正顯，室田一雄：線形計算の数理，岩波書店 (2009).

[12] Tamstorf, R., Jones, T. and McCormick, S.F.: Smoothed Aggregation Multigrid for Cloth Simulation, *ACM Trans. Graph.*, Vol.34, No.6, pp.245:1–245:13 (2015).

[13] Baraff, D. and Witkin, A.: Large Steps in Cloth Simulation, *Proc. 25th Annual Conference on Computer Graphics and Interactive Techniques*, *SIGGRAPH '98*, pp.43–54 (1998).

[14] Brezina, M. and Vassilevski, P.S.: Smoothed aggregation spectral element agglomeration AMG: SA-ρAMGe, *Large-Scale Scientific Computing*, LNCS, Vol.7116, pp.3–15 (2012).

[15] AMGS ライブラリ，入手先 ⟨http://hpcl.info.kogakuin.ac.jp/lab/software/amgs⟩.

[16] Hestenes, M.R. and Stiefel, E.: Methods of Conjugate Gradients for Solving Linear Systems, *Journal of Research of the National Bureau of Standards*, Vol.49, No.6, pp.409–436 (1952).

野村 直也　（正会員）

2015年工学院大学情報学部コンピュータ科学科卒業．2017年同大学大学院工学研究科情報学専攻修士課程修了．同年より東京大学大学院情報理工学系研究科数理情報学専攻博士課程に在学．大規模線形問題に対する Multigrid 法に興味を持つ．

中島 研吾　（正会員）

1985年東京大学工学部航空学科卒業，1993年テキサス大学オースティン校大学院航空宇宙工学・応用力学科修士課程修了，三菱総合研究所，高度情報科学技術研究機構（RIST），東京大学大学院理学系研究科を経て2008年より東京大学情報基盤センター教授．博士（工学）．専門は計算力学，数値線形代数，並列アルゴリズム．2018年より理化学研究所計算科学研究センター副センター長を兼務．日本計算工学会，日本応用数理学会，SIAM，IEEE 各会員．

藤井 昭宏　（正会員）

1999 年東京大学理学部情報科学科卒業．2001 年同大学大学院理学系研究科修士課程修了，2004 年同大学院情報理工学系研究科博士課程修了．工学院大学 CPD センターを経て，工学院大学大学院工学研究科准教授．博士（情報理工学）．数値線形代数，並列アルゴリズムの研究に従事する．IEEE-CS，電子情報通信学会各会員．