近似逆行列の並列計算による GMRES(m)法の高速化

平成19年度

張　臨傑

目 次

第 1 章

序 論

1.1 背 景

様々な自然現象を記述する偏微分方程式の境界値問題の数値解を求めるには，有限要素法や有限差分法で離散化して得られる大型で疎な行列を係数に持つ連立 1 次方程式

$$A\boldsymbol{x} = \boldsymbol{b}, \quad A \in \boldsymbol{R}^{n\times n}, \quad \boldsymbol{x}, \boldsymbol{b} \in \boldsymbol{R}^{n} \tag{1.1}$$

の解 $\boldsymbol{x}$ を求めることになる．

連立 1 次方程式 (1.1) の解法は，通常直接法と反復法に分類される．直接法には，ガウス消去法 [49] などがあり，演算に丸め誤差がなければ有限回の演算によって厳密解を得ることができる．しかし，連立 1 次方程式 (1.1) の次元が大きいときには，計算量が膨大になったり，フィルイン (0 であったところが，0 でなくなる現象) によって，余計な記憶容量を要求することになる．このため，大規模行列問題には，通常反復法が利用される．反復法とは，初期ベクトル $\boldsymbol{x}_0$ から出発して，適当な漸化式によって

$$\lim_{k\to\infty} \boldsymbol{x}_k = \boldsymbol{x}$$

を満たす $\boldsymbol{x}_k$ を順次に生成して，$\boldsymbol{x}_k$ が真の解 $\boldsymbol{x}$ に実用上十分に近づいたところで反復を打ち切り，$\boldsymbol{x}_k$ を $\boldsymbol{x}$ の近似解とする方法である．

反復法には，定常反復法と非定常反復法がある．定常反復法とは

$$\boldsymbol{x}_{k+1} = M\boldsymbol{x}_k + \boldsymbol{c}$$

の形で真の解 $\boldsymbol{x}$ に収束する列 $\boldsymbol{x}_k$ を生成する算法のことである．ここで，M は n 次の正方行列であり，$\boldsymbol{c}$ は n 次元ベクトルである．ただし，M と $\boldsymbol{c}$ は k に依存しない．非定常反復法とは

$$\boldsymbol{x}_{k+1} = M\boldsymbol{x}_k + \alpha_k\boldsymbol{p}_k$$

で $\boldsymbol{x}_k$ を生成する算法のことである．$\boldsymbol{p}_k$ は n 次元ベクトルであり，探索ベクトルとも呼ばれる．α_k はスカラーである．$\boldsymbol{p}_k$ と α_k ともに反復回数 k に依存する．定常反復法と比べて，非定常反復法は反復する時に，新しい情報を取り入れるので，定常反復法よりも少ない反復回数で近似解を求めることができる．

非定常反復法において，現在クリロフ部分空間法が主流となっている．クリロフ部分空間法とは，与えられた行列 $A \in \boldsymbol{R}^{n\times n}$ と初期残差ベクトル $\boldsymbol{r}_0 \in \boldsymbol{R}^n$ に対して，クリロフ部分空間

$$\mathcal{K}_m(A, \boldsymbol{r}_0) = \mathrm{span}\{\boldsymbol{r}_0, A\boldsymbol{r}_0, \ldots, A^{(m-1)}\boldsymbol{r}_0\}, m = 1, 2, \ldots, n \tag{1.2}$$

に属する近似解 $\boldsymbol{x}_m$ を求める算法である．係数行列 A が対称かつ正定値行列である場合，共役勾配法 (CG 法) [22] はもっとも効率の良い方法である．しかし，係数行列 A が非対称のときには，CG 法のように，簡単な 3 項間漸化式でクリロフ部分空間の正規直交基底を求めることができず，決定的な方法はまだ発見されていない．

大型非対称問題の場合，ランチョス過程 [27] に基づく BiCG 法 [17] (双共役勾配法) とアーノルディ過程 [1] に基づく GMRES (m) [42] 法が現在良く利用される．

BiCG 法は，互いに双直交する 2 つのクリロフ部分空間を生成し，2 つの異なる 3 項間漸化式でベクトルの直交性を保つ方法である．しかし，残差ノルムが常に減少する保証がないため，残差ノルムの収束が停滞したり，発散したりすることがある．残差ノルムの収束の安定性を良くするために，様々な研究が行われている [45, 40, 38]．

BiCG 法のこの欠点を解消するアーノルディ過程に基づく GMRES 法はクリロフ部分空間 $\mathcal{K}_m\ (A, \boldsymbol{r}_0)$ から，残差ベクトルの 2 ノルムを最小になるように，近似解 $\boldsymbol{x}_m$ を計算する．残差ノルムの減少は保証されている．GMRES 法は，精密演算では，理論的に高々 n 回の反復で精密解が得られることが知られている．しかし，GMRES 法ではクリロフ部分空間の正規直交基底はアーノルディ法によって計算されているので，CG 法のように簡単な 3 項間漸化式で求めることができない．このため，クリロフ部分空間の次元が増えるに伴って，正規直交基底の計算時間と必要となる記憶容量が増大する．GMERS 法をそのまま計算機に実装するのは実際不可能な場合が多い．通常は，計算量や記憶容量などを考慮してリスタート版の GMRES (m) 法を用いることが多い．

リスタート版の GMRES (m) 法は m 回反復して，計算した近似解 $\boldsymbol{x}_m$ を新たな初期ベクトルとし，リスタートし，再度直交化を行う．クリロフ部分空間の最大次元を m に制限したことによって，直交化のコストは軽減される．GMRES 法と比べて，リスタート型の GMRES (m) 法は記憶容量が少なくてすみ，実用的な利点を持っている．しかし，リスタートを行うことが原因で，近似解を構成する係数行列 A の固有ベクトルの情報が失われるこ

とがある．このため，残差ノルムの収束が停滞し，適切な計算時間内で収束しないこともある．残差ノルムの収束はリスタート周期 m に依存し，GMRES 法のように，高々 n 回の反復で収束する保証がない．一般にリスタート周期 m はユーザの経験によって決められる．

リスタート周期を決定するさまざまな研究が行われている．津野ら [14] は GMRES (m) 法の残差ノルムの収束する原理を残差多項式のゼロ点分布の観点から考察し，リスタート周期を自動的に早める手法を提案している．この手法は，残差多項式のゼロ点を複素平面上でなるべく集中させないことで，未知の理想的なゼロ点，つまり係数行列の固有値の分布に重なる可能性を高めることを現実的な目標としている．この手法でリスタートの判断をするには，GMRES(m) 法の残差多項式の係数，ゼロ点の分布が集中であるかどうかの条件判定を計算する必要がある．また，森屋ら [31, 32] は野寺ら [13, 39] が提案している ORTHOMIN(k) 法の適応的なリスタート手法の考えに基づいて，GMRES(m) 法の改良版の 1 つである DEFLATED-GMERS(m,k) 法のリスタート周期を動的に切り替える算法を提案している．この算法は，通常 DEFLATED-GMERS(m,k) 法を短いリスタート周期で反復させ，残差ノルムの収束が停滞したときにのみ，リスタート周期を変化させる．残差ノルムの収束の停滞判定基準は残差ベクトルと探索ベクトルのなす角度で規定される．さらに，羽部ら [21] は残差ノルムの減少の割合に応じて，リスタート周期の値を増減させる手法を提案している．

本論文では，上記手法と違って，係数行列の近似固有値である Ritz 値と調和 Ritz 値と残差ノルムの収束との関係に注目して，GMRES(m) 法の適応的なリスタート手法を提案する．提案する手法は，Ritz 値と調和 Ritz 値の差が大きければ，GMRES(m) 法の残差ノルムの収束が遅くなる傾向があると考え，適応的にリスタートを行う．リスタートの判断を行うための Ritz 値と調和 Ritz 値は低いコストで計算でき，実装しやすい算法である．

また，式 (1.1) に GMRES (m) 法を直接適用すると，実用上，真の解に十分に近づいた近似解を求めるまでの反復回数が膨大になることがある．このため，GMRES (m) 法を適用する前に，行列の前処理を行う手法が良く使われている．行列の前処理によって，近似解が求めるまでの反復回数を大幅に減少させることができる．行列の前処理とは，式 (1.1) に対する次式のような行列変換を行うことである．すなわち，行列 $M \in \boldsymbol{R}^{n\times n}$ を計算し，式 (1.1) の係数行列 A の右側から掛けて

$$\begin{cases} AM\boldsymbol{y} = \boldsymbol{b} \\ \boldsymbol{x} = M\boldsymbol{y} \end{cases}$$

もしくは左側から掛けて，

$$MA\boldsymbol{x} = M\boldsymbol{b}$$

のように係数行列の変換をして近似解を計算する手法である．前処理行列 M は，AM が A より条件数が小さいようなものを選ぶ．一般的に言うと，AM が単位行列 I_n に近いほうが望ましい．また，AM の固有値が原点から離れた所で群がる場合にも良い収束が得られる．このような小さい条件数と良い固有値分布のほか，前処理行列 M が簡単に求められることと低いコストで実装できることも必要条件の 1 つである．

前処理行列 M の選択について，現在では様々な方法が提案されているが，主流は直接法から得られた係数行列 A の近似LU 分解に基づく前処理である．例えばILU 分解をあげることができる [10]．ILU 分解とは，係数行列 A を下三角行列 L と上三角行列 U の積に近似分解し，$(LU)^{-1}$ を前処理行列 M にする前処理技法である．この場合，連立 1 次方程式 $LU\boldsymbol{v} = \boldsymbol{u}$ を解くことによって，前処理行列 $M = (LU)^{-1}$ とベクトル $\boldsymbol{u}$ との積 $\boldsymbol{v}$ を求めることになる．行列 L と U は三角行列なので，方程式 $LU\boldsymbol{v} = \boldsymbol{u}$ を前進代入と後退代入で簡単に解ける．しかし，ILU 分解による行列の前処理は並列化しにくいと言う難点がある．また，係数行列の非ゼロ要素の分布が不規則の場合，コストが高くなることもある．

並列計算の領域では，係数行列 A の不完全行列分解の代わりに，A の近似逆行列を求める前処理手法も提案されている．このタイプの前処理は前処理行列となる近似逆行列を明示的に計算，保存するため，前処理行列 M とベクトルの積は行列とベクトルの積に帰着するので，実装しやすい．その並列計算も簡単にできる．さらに，不規則な非ゼロ要素の分布を持つ一般行列にも簡単に適用できるため，注目を集めている．現在では，行列 A の近似逆行列を求めるために，様々な方法が提案されている．例えば，$||I_n - AM||$ を最小にすることによって，係数行列 A の近似逆行列 M を求める MR 法 [19] がある．この手法は前処理行列 M の各列をまったく独立に計算することができるので，非常に並列化しやすい方法である．さらに，最近，Sherman-Morrision 公式による近似逆行列法 (AISM 法) [8] も提案されている．AISM 法は係数行列 A の逆行列を

$$s^{-1}I_n - s^{-2}U\Omega^{-1}V^T$$

で近似する算法である．ここで，s はスカラーであり，U と V は n 次元の正方一般行列であり，Ω は n 次元の正方対角行列である．MR 法と比べて，AISM 法は多くの場合，前処理としての性能が安定していると観察されている．しかし，係数行列の次元が大きい場合，その近似逆行列を計算するために，膨大な時間が掛かるケースがしばしばある．しかも，計算過程にベクトルどうしの依存関係が存在しているため，並列計算に不向きである．このため，前処理のコストが高くなり，大型問題の場合，採用されないことが多い．

本論文では，AISM 法の前処理としての安定性に注目し，AISM 法を大型問題にも適用できるようにするために，2 つの並列手法を提案する．

1.2 目 的

本論文では，大型で非対称，疎な連立1次方程式を解くためのリスタート版のGMRES(m)法の高速化のために，次の手法を提案する．

- Ritz 値と調和 Ritz 値による適応的なリスタート手法
- Naik[36] による並列実装法を利用した AISM 法の部分並列化
- AISM 法を用いた 2 レベル並列計算

Ritz 値と調和 Ritz 値は行列の近似固有値である．Ritz ベクトルと調和 Ritz ベクトルは，近似固有ベクトルである．近年，これらは前処理の観点から利用されることが多い．例えば，MORGAN(m,k) 法 [29, 30]，DEFLATED-GMRES(m,k) 法 [16] などがある．本論文では，上記算法と違って，リスタート周期 m を適応的な変えるために，Ritz 値と調和 Ritz 値を利用する．即ち，GMRES(m) 法のリスタートするタイミングを Ritz 値と調和 Ritz 値を使って決める．リスタート周期 m を設定するユーザの負担を軽減し，不必要な直交化処理を削減することで収束に必要な計算時間を短縮することを目的としている．

AISM 法は Sherman-Morrision 公式を利用して，一般行列の疎な近似逆行列を計算する方法である．MR 法，ILU 分解と比べて，AISM 法は多くの場合，前処理としての性能が安定していることが観察されている．しかし，係数行列の次元が大きい場合，近似逆行列の計算時間が膨大になる可能性がある．しかも，計算過程にベクトルどうしの依存関係が存在しているため，並列計算機での計算にも不向きである．このため，大型問題の場合，採用されないことが多い．本論文では，AISM 法を大型問題にも適用できるようにするために，2 つの並列手法を提案する．

1 つは Naik [36] の実装を利用した部分並列化である．Naik [36] は，Benchmark BT という非優位対角な 5×5 のブロックを持つ 3 重ブロック対角行列を係数とする線形システムを LU 分解で解く過程をパイプライン制御により部分的に並列化している．本論文では，Naik の 1 次元分割の実装法を AISM 法による近似逆行列の計算に適用し，AISM 法を部分的に並列化する手法を提案する．

他の 1 つは，AISM 法を用いた 2 レベル並列計算である．2 レベル並列計算とは，部分構造法の観点に立ち，グラフ分割や領域分割などを利用して，係数行列 A を並列計算しやす

い形式

$$\begin{bmatrix} A_1 & & & B_1 \\ & \ddots & & \vdots \\ & & A_p & B_p \\ C_1 & \cdots & C_p & A_s \end{bmatrix}$$

に再構成し，その近似逆行列を並列計算で求める並列手法である．本論文では，この手法に AISM 法を適用して，近似逆行列の並列計算を提案する．

1.3 本論文の構成

本論文では，まず，第 2 章で，GMRES(m) 法について簡単に述べる．次に，第 3 章において，Ritz 値と調和 Ritz 値による GMRES(m) 法の適応的なリスタートを提案する．第 4 章では，Sherman-Morrison 法による近似逆行列法 (AISM 法) を中心に，行列の前処理について述べる．第 5 章では，Naik [36] の実装法に基づく部分並列化を提案する．その後，第 6 章において，AISM 法を用いた 2 レベル並列計算について述べる．最後に第 7 章で，総括と結論を述べる．

第 2 章

GMRES(m)法

式 (1.1) を解くための反復法の 1 つとして，また固有値問題の解法として，クリロフ部分空間法はよく使われる．GMRES 法はそのような解法の 1 つである．クリロフ部分空間は，与えられた行列 $A \in \boldsymbol{R}^{n\times n}$ とベクトル $\boldsymbol{r}_0 \in \boldsymbol{R}^n$ に対して，次のように定義される部分空間のことである．

$$\mathcal{K}_m(A, \boldsymbol{r}_0) = \mathrm{span}\{\boldsymbol{r}_0, A\boldsymbol{r}_0, \ldots, A^{(m-1)}\boldsymbol{r}_0\}, \quad m = 1, 2, \ldots, n. \tag{2.1}$$

GMRES 法はこのクリロフ部分空間から近似解を残差の 2 ノルムを最小にするように求める算法である．近似解はクリロフ部分空間の直交基底によって構成される．通常，このクリロフ部分空間の正規直交基底は，図 2.1 に示すようなアーノルディ法 [1] により生成される．ベクトル列 $(\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_m)$ は，生成したクリロフ部分空間 $\mathcal{K}_m(A, \boldsymbol{r}_0)$ の正規直交基底である．また，この正規直交基底 $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_m\}$ は，アーノルディの正規直交基底とも呼ばれている．今，スカラー $H_m(i, j) \equiv h_{i,j}$ とすると，$H_m \in \boldsymbol{R}^{m\times m}$ は，上ヘッセンベルグ行列となる．行列 $V_m = (\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_m) \in \boldsymbol{R}^{n\times m}$ の列ベクトルは，クリロフ部分空間 $\mathcal{K}_m(A, \boldsymbol{r}_0)$ の正規直交基底なので，行列 V_m は正規直交行列である．行列 H_m と行列 V_m は，次の関係を満たす．

$$\begin{aligned} AV_m &= V_m H_m + h_{m+1,m}\boldsymbol{v}_{m+1}\boldsymbol{e}_m^H \\ &= V_{m+1}\bar{H}_m \end{aligned} \tag{2.2}$$

ただし， $\bar{H}_m \in \boldsymbol{R}^{(m+1)\times m}$ は，上ヘッセンベルグ行列 H_m の最後の一行の下に，行ベクトル $(0 \ldots 0\ h_{m+1,m})$ を加えた行列である．$\boldsymbol{e}_m \in \boldsymbol{R}^n$ は，m 番目の要素が 1 の単位ベクトルである．ここで，式 (2.2) に左から V_m^H を掛けると

$$H_m = V_m^H A V_m \tag{2.3}$$

となる．クリロフ部分空間 $\mathcal{K}_m(A, \boldsymbol{r}_0)$ から，GMRES 法の近似解 $\boldsymbol{x}_m$ は次のように計算することになる．

$$\boldsymbol{x}_m = \boldsymbol{x}_0 + V_m\boldsymbol{y}_m, \quad \boldsymbol{y}_m = \min_{\boldsymbol{y}} \| \gamma\boldsymbol{e}_1 - \bar{H}_m\boldsymbol{y} \|_2 \tag{2.4}$$

1 : **Start with** A **and** $\boldsymbol{r}_0$
2 : $\boldsymbol{v}_1 = \frac{\boldsymbol{r}_0}{\|\boldsymbol{r}_0\|}$
3 : **for** $j = 1, 2, \ldots, m$
4 : $\quad \bar{\boldsymbol{v}} = A\boldsymbol{v}_j$
5 : $\quad$ **for** $i = 1, 2, \ldots, j$
6 : $\quad\quad h_{i,j} = \boldsymbol{v}_i^H \bar{\boldsymbol{v}}$
7 : $\quad\quad \bar{\boldsymbol{v}} = \bar{\boldsymbol{v}} - h_{i,j}\boldsymbol{v}_i$
8 : $\quad$ **end for**
9 : $\quad h_{j+1,j} = \| \bar{\boldsymbol{v}} \|_2$
10: $\quad \boldsymbol{v}_{j+1} = \frac{\bar{\boldsymbol{v}}}{h_{j+1,j}}$
11: **end for**

図 2.1 アーノルディ法

ただし，$\boldsymbol{x}_0$ は初期ベクトルであり，$\gamma = \| \boldsymbol{r}_0 \|_2$，$\boldsymbol{e}_1 \in \boldsymbol{R}^n$ は 1 番目の要素が 1 の単位ベクトルである．通常，Givens 回転を利用して，行列 H_m を直交行列と上三角行列に分解することによって，$\boldsymbol{y}_m$ を求めることになる．この時の残差の 2 ノルムは

$$\| \boldsymbol{r}_m \|_2 = \| b - A\boldsymbol{x}_m \|_2 = h_{m+1,m} |\boldsymbol{e}_m^T \boldsymbol{y}_m| \tag{2.5}$$

となる．図 2.2 に GMRES 法のアルゴリズムを示す．GMRES 法に関する詳細については，Saad ら [42] を参照のこと．

GMRES 法によって，理論的には高々 n 回の反復で解を求めることが可能である．しかし，GMRES 法は非対称行列を扱うため，CG 法のように簡単な 3 項間漸化式で書き表すことができない．反復回数が増えるたびに，ベクトル列 $(\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_m)$ の計算コストが高くなる．しかも上ヘッセンベルグ行列 $\bar{H}_m$ を保存しておく必要がある．このことは，計算量および記憶容量の点から実用的であるとは言えない．そこで，直交ベクトルの最大本数を $m (\ll n)$ 本に制限し，そのときの近似解を新たな初期近似解として次の反復を行うリスタート版の GMRES(m) 法がよく使われる．そのアルゴリズムを図 2.3 で示す．m はリスタート周期と呼ばれる．本章では，リスタートなしの GMRES 法を GMRES 法と呼ぶ．また，リスタート周期を固定する GMRES(m) 法を古典的な GMRES(m) 法と呼ぶ．

1. choose an initial guess x_0
2. set $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$, $\gamma = \| \boldsymbol{r}_0 \|_2$
3. $\boldsymbol{v}_1 = \boldsymbol{r}_0/\gamma$
4. for $n = 1, 2, \ldots$
5. $\quad \hat{\boldsymbol{v}} = A\boldsymbol{v}_n$
6. $\quad$ for $i = 1, 2, \ldots, n$
7. $\quad\quad h_{i,n} = (\hat{\boldsymbol{v}}, \boldsymbol{v}_i)$
8. $\quad\quad \hat{\boldsymbol{v}} = \hat{\boldsymbol{v}} - h_{i,n}\boldsymbol{v}_i$
9. $\quad$ end
10. $\quad h_{n+1,n} = \| \hat{\boldsymbol{v}} \|_2$
11. $\quad \boldsymbol{v}_{n+1} = \hat{\boldsymbol{v}}/h_{n+1,n}$
12. $\quad \boldsymbol{y} = \min_{\boldsymbol{y}} \| \gamma \boldsymbol{e}_1 - \overline{H}_n \boldsymbol{y} \|_2$
13. $\quad \boldsymbol{x}_n = \boldsymbol{x}_0 + V_m \boldsymbol{y}, \boldsymbol{r}_n = \boldsymbol{b} - A\boldsymbol{x}_n$
14. $\quad$ if $\boldsymbol{r}_n < \varepsilon$ then
15. $\quad\quad$ stop iteration
16. $\quad$ endif
17. end

図 2.2 GMRES 法

1. choose an initial guess $\boldsymbol{x}_0$
2. set $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$, $\gamma = \| \boldsymbol{r}_0 \|_2$
3. $\boldsymbol{v}_1 = \boldsymbol{r}_0/\gamma$
4. for $n = 1, 2, \ldots, m$
5. $\quad \hat{\boldsymbol{v}} = A\boldsymbol{v}_n$
6. $\quad$ for $i = 1, 2, \ldots, n$
7. $\quad\quad h_{i,n} = (\hat{\boldsymbol{v}}, \boldsymbol{v}_i)$
8. $\quad\quad \hat{\boldsymbol{v}} = \hat{\boldsymbol{v}} - h_{i,n}\boldsymbol{v}_i$
9. $\quad$ end
10. $\quad h_{n+1,n} = \| \hat{\boldsymbol{v}} \|_2$
11. $\quad \boldsymbol{v}_{n+1} = \hat{\boldsymbol{v}}/h_{n+1,n}$
12. $\quad \boldsymbol{y} = \min_{\boldsymbol{y}} \| \gamma \boldsymbol{e}_1 - \overline{H}_n \boldsymbol{y} \|_2$
13. $\quad \boldsymbol{x}_n = \boldsymbol{x}_0 + V_m \boldsymbol{y}, \boldsymbol{r}_n = \boldsymbol{b} - A\boldsymbol{x}_n$
14. $\quad$ if $\boldsymbol{r}_n < \varepsilon$ then
15. $\quad\quad$ stop iteration
16. $\quad$ endif
17. end
19. set $\boldsymbol{r}_0 = \boldsymbol{r}_m$
20. go to 2

図 2.3 GMRES(m) 法

第 3 章

Ritz値と調和Ritz値によるGMRES(m)法の高速化

GMRES 法と比べて，GMRES(m) 法は要求する記憶容量が少なくてすみ，実用的な利点を有す．しかし，その収束性はリスタート周期 m に大きく影響される．一般に，GMRES(m) 法のリスタート周期 m はユーザの経験によって決めることになる．リスタート周期を決定するさまざまな手法 [14, 13, 31, 39, 32, 21] も提案されているが，現在のところ m を決定する最適な方法はまだ提案されていない．

Ritz 値と調和 Ritz 値は行列の近似固有値である．また，Ritz ベクトルと調和 Ritz ベクトルは，係数行列 A の近似固有ベクトルである．近年，係数行列の近似固有値，近似固有ベクトルを利用して，GMRES 法の残差ノルムの収束性を改善する様々な方法が提案されている．例えば，MORGAN(m,k) 法 [29, 30]，DEFLATED-GMRES(m,k) 法 [16] などがある．MORGAN(m,k) 法は，リスタートする際に，既存のクリロフ部分空間から絶対値の小さい k 個の近似固有値に対応する近似固有ベクトルを，次のクリロフ部分空間に基底ベクトルとして追加する算法である．DEFLATED-GMRES(m,k) 法は，リスタートする際に，k 個の Ritz ベクトルを求め，前処理を行う算法である．ただし，これらの算法は，リスタート周期 m とパラメータ k を指定する必要がある．なお，Moriya ら [31] は，野寺ら [39, 13] が提案した ORTHOMIN(k) 法の適応的なリスタート手法の考えに基づいて，DEFLATED-GMRES(m) 法のリスタート周期を自動的に決定する算法を提案した．

本論文では，上記算法と違って，Ritz 値と調和 Ritz 値を使って，リスタート周期 m を動的に変えるリスタート手法を提案する．即ち，GMRES(m) 法のリスタートするタイミングを Ritz 値と調和 Ritz 値を使って決定する．この算法を使うと，古典的な GMRES(m) 法と比べて，リスタート周期 m を設定するユーザの負担が軽減され，不必要な直交化処理を削減することで収束に必要な計算時間が短縮できることが期待される．

まず，第 3.1 節では，Ritz 値と調和 Ritz 値について述べる．第 3.2 節では，Ritz 値と調和 Ritz 値と GMRES 法の残差ノルムの収束性との関係について述べ，Ritz 値と調和 Ritz 値を考慮した GMRES(m) 法の適応的なリスタート手法を提案する．第 3.3 節では，数値実

験について述べ，このリスタート手法の有効性を示す．第 3.4 節では，本章のまとめと結論を述べる．

3.1 Ritz 値と調和 Ritz 値

定義 1(Sleijpen ら [46]) 線形部分空間 $\mathcal{W} \subset \boldsymbol{C}^n$ に対して

$$\boldsymbol{x} \in \mathcal{W}, \quad \boldsymbol{x} \neq 0, \quad (A\boldsymbol{x} - \mu\boldsymbol{x}) \quad \perp \quad \mathcal{W} \tag{3.1}$$

を満たすベクトル $\boldsymbol{x}$ を行列 A の部分空間 $\mathcal{W}$ に対する Ritz ベクトルと呼び，スカラー μ を Ritz ベクトル $\boldsymbol{x}$ に対する Ritz 値とする．

この定義に従って，係数行列 A のクリロフ部分空間 $\mathcal{K}_m(A, \boldsymbol{r}_0)$ に対する Ritz 値 μ と Ritz ベクトル $\boldsymbol{x}$ を求めることを考える．

$$(A\boldsymbol{x} - \mu\boldsymbol{x}) \quad \perp \quad \mathcal{K}_m(A, \boldsymbol{r}_0) \Leftrightarrow V_m^H(AV_m\boldsymbol{y}_m - \mu V_m\boldsymbol{y}_m) = 0 \tag{3.2}$$

ただし，

$$\boldsymbol{x} = V_m\boldsymbol{y}_m, \quad \boldsymbol{x} \in \mathcal{K}_m(A, \boldsymbol{r}_0), \quad \boldsymbol{y}_m \in \boldsymbol{R}^m.$$

ここで，V_m は式 (2.2) の V_m と一致する．式 (3.2) に式 (2.3) を代入すると

$$H_m\boldsymbol{y}_m = \mu\boldsymbol{y}_m \tag{3.3}$$

となる．従って，クリロフ部分空間 $\mathcal{K}_m(A, \boldsymbol{r}_0)$ から選んだ Ritz ベクトル $\boldsymbol{x} = V_m\boldsymbol{y}_m$ に対応する Ritz 値は行列 H_m の固有値である．ベクトル $\boldsymbol{y}_m$ は行列 H_m の固有ベクトルである．つまり，次元の小さい行列 H_m の固有値問題を解くことによって，行列 A の Ritz 値と Ritz ベクトルは得られることになる．本論文では，Ritz ベクトルを使わないので Ritz 値だけを計算すれば良い．

定義 2(Sleijpen ら [46]) 線形部分空間 $\mathcal{W} \subset \boldsymbol{C}^n$ に対して

$$\boldsymbol{x} \in \mathcal{W}, \ \boldsymbol{x} \neq 0, \ (A^{-1}\boldsymbol{x} - \mu\boldsymbol{x}) \quad \perp \quad \mathcal{W} \tag{3.4}$$

を満たすベクトル $\boldsymbol{x}$ を行列 A の部分空間 $\mathcal{W}$ に対する調和 Ritz ベクトルと呼び，スカラー $\bar{\mu} = 1/\mu$ を調和 Ritz ベクトル $\boldsymbol{x}$ に対する調和 Ritz 値とする．

本論文では，調和 Ritz ベクトルを使わないので，調和 Ritz 値の計算についてのみ考える．調和 Ritz 値を求める際に，A^{-1} の計算を避けるために，下記の定理を使用する．

定理 1(Sleijpen ら [46]) $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$ を m 次元の部分空間 $\mathcal{V}_m$ の正規直交基底とする．スカラー $\bar{\mu}$ は行列 A の部分空間 $\mathcal{W}_m = A\mathcal{V}_m$ に対する調和 Ritz 値であることは

$$A\boldsymbol{x} - \bar{\mu}\boldsymbol{x} \perp A\mathcal{V}_m, \quad \boldsymbol{x} \in \mathcal{V}_m, \quad \boldsymbol{x} \neq 0 \tag{3.5}$$

が成立することと同値である．証明については，文献 [46] を参照してほしい．

本論文では，この定理に基づいて部分空間 $A\mathcal{K}_m(A, \boldsymbol{r}_0)$ に対する A の調和 Ritz 値を計算する．まずクリロフ部分空間 $\mathcal{K}_m(A, \boldsymbol{r}_0)$ から，ベクトル $\boldsymbol{x} = V_m\boldsymbol{y}_m$ を選んで，式 (3.5) に代入すると

$$\begin{aligned} (A\boldsymbol{x} - \bar{\mu}\boldsymbol{x}) \quad &\perp \quad A\mathcal{K}_m(A, \boldsymbol{r}_0) \\ &\Leftrightarrow \quad (AV_m)^H(AV_m\boldsymbol{y}_m - \bar{\mu}V_m\boldsymbol{y}_m) = 0 \end{aligned}$$

となる．さらに，式 (2.2) を利用すると

$$\bar{H}_m^H\bar{H}_m\boldsymbol{y}_m = \bar{\mu}H_m^H\boldsymbol{y}_m \tag{3.6}$$

となる．式 (3.6) に左から H_m^{-H} を掛けると，次式を得る．

$$H_m^{-H}\bar{H}_m^H\bar{H}_m\boldsymbol{y}_m = \bar{\mu}\boldsymbol{y}_m \tag{3.7}$$

今，$H_m^{-H}\bar{H}_m^H\bar{H}_m$ を整理すると，

$$H_m^{-H}\bar{H}_m^H\bar{H}_m = H_m + h_{m+1,m}^2\boldsymbol{f}_m\boldsymbol{e}_m^H \tag{3.8}$$

となる．ただし，$\boldsymbol{f}_m = H_m^{-H}\boldsymbol{e}_m$ である．ここで，行列 $H_m + h_{m+1,m}^2\boldsymbol{f}_m\boldsymbol{e}_m^H$ の固有値は，行列 A の空間 $A\mathcal{K}_m(A, \boldsymbol{r}_0)$ に対する調和 Ritz 値であることがわかる．また，$\boldsymbol{f}_m$ を計算するためには，H_m^{-H} を計算する必要があるが，この計算コストは A^{-1} の計算コストよりはるかに低いので，本論文では，この手法を利用して，調和 Ritz 値を求めることにする．

3.2 Ritz 値と調和 Ritz 値による適応的なリスタート

本節では，まず，Ritz 値と調和 Ritz 値と GMRES 法の収束との関係について述べる．その次に，Ritz 値と調和 Ritz 値による GMRES 法の適応的なリスタート手法を提案する．

Ritz 値と GMRES 法の収束との関係

第 3.1 節では，Ritz 値は H_m の固有値であることと，調和 Ritz 値は $\mathcal{H}_m = H_m + h_{m+1,m}^2\boldsymbol{f}_m\boldsymbol{e}_m^H$ の固有値であることについて述べた．不変なクリロフ部分空間が得られた場合には，つま

り新しく加えたベクトルが既存のクリロフ部分空間の直交基底の線形結合である場合には，図 2.1 に示すアーノルディ法の算法から $h_{m+1,m} = 0$ は明らかとなる．即ち，Ritz 値と調和 Ritz 値は完全に一致することになる．そうではない場合，Ritz 値と調和 Ritz 値の差について，$\| h_{m+1,m}^2 \boldsymbol{f}_m \boldsymbol{e}_m^H \|$ を考えばよい．ここで，$\boldsymbol{f}_m = H_m^{-H} \boldsymbol{e}_m$ なので，次式が成立する．

$$\begin{aligned} \| \boldsymbol{f}_m \| &= \| H_m^{-H} \boldsymbol{e}_m \| \\ &\leq 1/\sigma_{\min}(H_m) \end{aligned}$$

ただし，$\sigma_{\min}(H_m)$ は H_m の最小な特異値である．故に，Ritz 値と調和 Ritz 値の差の上界は

$$\frac{h_{m+1,m}^2}{\sigma_{\min}(H_m)} \tag{3.9}$$

となる [18]．従って，Ritz 値と調和 Ritz 値の差が大きければ，$|h_{m+1,m}|$ の値が大きいか，または $\sigma_{\min}(H_m)$ の値が小さいことになる．式 (2.5) から，$|h_{m+1,m}|$ の値が大きければ，残差ノルム $\| \boldsymbol{r}_m \|$ の値が大きいことがわかる．$\sigma_{\min}(H_m)$ の値が小さければ，式 (2.4) の $\boldsymbol{y}_m$ の計算は丸め誤差に影響されやすいことになる．これは GMRES 法の残差ノルムが停滞する場合と一致する．即ち，Ritz 値と調和 Ritz 値の差が大きければ，GMRES 法の残差ノルムが停滞することを意味している．よって，Ritz 値と調和 Ritz 値の差から，GMRES 法の残差ノルムの収束を評価することができる．

上記考察に基づいて GMRES(m) 法の適応的なリスタート手法を提案する．

Ritz 値による適応的なリスタート

この算法では，反復するたびに，Ritz 値と調和 Ritz 値の差を計算する．もしその差が一つ前の反復における差より大きければ，GMRES 法の残差ノルムの収束が悪くなる傾向があると考える．そこで，残差ノルムの停滞を避けるために，現在のクリロフ部分空間から近似解を計算し，それを初期値として，新しいクリロフ空間を生成する．つまり，リスタートを行うことになる．そうでない場合，リスタートせずに計算を続行する．このようなリスタートを行うことで，不必要な反復を削減でき，収束に必要な計算時間が短縮できることが期待される．ただし，問題によっては，Ritz 値と調和 Ritz 値の差が順調に減少してゆくと，クリロフ部分空間の次元がかなり大きくなっても，リスタートしない可能性もある．メモリ容量の制限から考えると，最大リスタート周期を指定する必要がある．それに，オプションとして，最小リスタート周期を指定することもできる．本稿では，最小リスタート周期を 1 にする．また，Ritz 値と調和 Ritz 値の差の計算について，本稿では，最大 Ritz 値と最大調和 Ritz 値の差の絶対値を利用する．

具体的に，まず，アーノルディ過程によって上ヘッセンベルグ行列 $\bar{H}_m$ と H_m を計算する．次に，行列 $\mathcal{H}_m = H_m + h_{m+1,m}^2 \boldsymbol{f}_m \boldsymbol{e}_m^H$ を計算する．ただし，$\boldsymbol{f}_m = H_m^{-H} \boldsymbol{e}_m$ とする．そ

の後，行列 H_m と $\mathcal{H}_m$ の最大固有値 $\mu_{\max}$ と $\bar{\mu}_{\max}$ を求めて，両者の差 $D_{cur} = |\mu_{\max} - \bar{\mu}_{\max}|$ を計算する．もしこの値が前の反復における差 D_{pre} より大きければ，近似解 $\boldsymbol{x}_m$ を算出し，新たな初期近似解 $\boldsymbol{x}_0$ としてリスタートする．本稿では，Ritz 値を考慮した GMRES(m) 法を RITZ-GMRES 法と呼び，その算法を図 3.1 に示す. ただし，$m_{\max}(\ll n)$ はユーザが設定する最大リスタート周期であり，$i_{\max}$ は最大反復回数である．また図 3.1 には，行列の下添字を省略している．

古典的な GMRES(m) 法と比べると，Ritz 値を考慮した RITZ-GMRES 法では，行列 $\mathcal{H}_m$ を記憶するための記憶容量が必要となる．ただし，要求される記憶容量はクリロフ部分空間の次元だけに依存するので，実際にはあまり問題にならない．

演算量の面から比較すると，RITZ-GMRES 法では $\boldsymbol{f}_m = H_m^{-H}\boldsymbol{e}_m$ を求めなければならない．それに，行列 H_m と $\mathcal{H}_m$ の最大固有値，つまり最大 Ritz 値と最大調和 Ritz 値を求めなければならない．しかし，先ほど述べたように，行列 H_m と $\mathcal{H}_m$ の次元はクリロフ部分空間の次元だけに依存する．実際には取り扱う行列はそれほど大きなものにはならない ($m_{\max}$ を超えない) ので，逆行列と固有値を低いコストで計算できる．

ここで，$\boldsymbol{f}_m = H_m^{-H}\boldsymbol{e}_m$ の計算には，CLAPACK[2] のルーチン dgesv_を利用した．dgesv_は一般行列を係数とする連立 1 次方程式を解く倍精度ルーチンである．連立 1 次方程式の係数行列を行列 H_m に設定して，方程式の右辺を同次元の単位行列に設定すると，求められる解は係数行列 H_m の逆行列となる．行列 H_m と $\mathcal{H}_m$ の固有値，つまり Ritz 値と調和 Ritz 値の計算に，CLAPACK のルーチン dgeev_を用いた．

3.3 数値実験

本節では，数値実験をとおして，提案したリスタート手法の有効性を評価することにする．計算機の仕様は下記のものを利用し，数値実験は以下の環境で行った．

計算機: DELL PowerEdge 1750

OS: Red Hat Linux 9.0

CPU: 3.00 GHz ×1 インテル (R) Xeon(R)

メモリ: 4GB

収束判定条件: $\| \boldsymbol{r}_m \|_2 \,/\, \| \boldsymbol{r}_0 \|_2 \leq 1.0 \times 10^{-12}$

初期値ベクトル: $\boldsymbol{x}_0 = (0, 0, \ldots, 0)^T$

1: **Choose an intial guess** $\boldsymbol{x}_0$**, set** $i = 1$**,** $m_{\min} = 1$**,** $m_{pre} = -1$**,**
2: **Start**
3: **Set** $m = 1$**,** $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$**,** $\beta = \| \boldsymbol{r}_0 \|_2$**,** $\boldsymbol{v}_1 = \boldsymbol{r}_0/\beta$
4: **While** $m \leq m_{\max}$
5: $\bar{\boldsymbol{v}} = A\boldsymbol{v}_m$
6: **For** $i = 1, 2, \ldots, m-1$
7: $\bar{H}(i,m) = \boldsymbol{v}^H \bar{\boldsymbol{v}}$
8: $\bar{\boldsymbol{v}} = \bar{\boldsymbol{v}} - \bar{H}(i,m)\boldsymbol{v}_i$
9: **End for**
10: $\bar{H}(m+1,m) = \| \bar{\boldsymbol{v}} \|_2$
11: $\boldsymbol{v}_{m+1} = \frac{\bar{\boldsymbol{v}}}{\bar{H}(m+1,m)}$
13: **Set** $H(i,j) = \bar{H}(i,j)$**,**$i,j = 1, \ldots, m$
14: **Compute** $\boldsymbol{f}_m = H^{-H}\boldsymbol{e}_m$ **and** $\mathcal{H} = H + h_{m+1,m}^2 \boldsymbol{f}_m \boldsymbol{e}_m^H$
15: **Compute eigenvalues** μ_l **of** H **and eigenvalues** $\bar{\mu}_l$ **of** $\mathcal{H}$
16: **Set** $D_{cur} = |\mu_{\max} - \bar{\mu}_{\max}|$
17: **If** $i > i_{\max}$
18: $\boldsymbol{y} = \min_y \| \beta\boldsymbol{e}_1 - \bar{H}\boldsymbol{y} \|_2$
19: $\boldsymbol{x}_m = \boldsymbol{x}_0 + V_m\boldsymbol{y}, \boldsymbol{r}_m = b - A\boldsymbol{x}_m$
20: **Stop iteration**
21: **If** $i > 1$
22: **If** ($D_{cur} > D_{pre}$ **and** $m >= m_{\min}$) **or** $m == m_{\max}$
23: $\boldsymbol{y} = \min_y \| \beta\boldsymbol{e}_1 - \bar{H}\boldsymbol{y} \|_2$
24: $\boldsymbol{x}_m = \boldsymbol{x}_0 + V_m\boldsymbol{y}, \boldsymbol{r}_m = b - A\boldsymbol{x}_m$
25: **If** $\| \boldsymbol{r}_m \|_2 < \epsilon$ **Stop iteration**
26: **Set** $\boldsymbol{x}_0 = \boldsymbol{x}_m$**,** $D_{pre} = D_{cur}$
27: **Go to start**
28: **Set** $D_{pre} = D_{cur}$**,** $m = m + 1$**,** $i = i + 1$
29: **End while**

図 3.1 Ritz 値を考慮した RITZ-GMRES($m_{\max}$) 法

最大反復回数: 20000

プログラム言語: C 言語

計算精度: 倍精度

本章で提案したリスタート手法の有効性を示すために，古典的な GMRES(m) 法との比較を中心に数値実験を行った．ただし，古典的な GMRES(m) 法を GMRES(m) で表記し，本論文で提案した RITZ-GMRES 法を RITZ-GMRES で表記する．また，最大 Ritz 値を $\mu_{\max}$ で表記し，最大調和 Ritz 値を $\bar{\mu}_{\max}$ で表記する．

RITZ-GMRES 法の最大リスタート周期 $m_{\max}$ を 50 に設定した．GMRES(m) 法のリスタート周期 m を m =10，20，30，40，50 に設定した．各実験において，RITZ-GMRES 法の実際のリスタート周期の最大値と平均値 (小数点以下 3 桁目を四捨五入) を計算し，それと一番近いリスタート周期を持つ古典的な GMRES(m) 法との比較を行った．

GMRES 法の反復回数については，アーノルディ過程の 1 反復につき 1 回と数えた．計算時間については Clock() 関数で求めた値を秒単位で表した．

[数値例 1] 正方領域 $\Omega = [0,1] \times [0,1]$ における 2 次元楕円型偏微分方程式の境界値問題を考える [25].

$$\begin{aligned} -u_{xx}(x,y) - u_{yy}(x,y) + D((y-1/2)u_x(x,y) + (x-1/3)(x-2/3)u_y(x,y))) &= G(x,y), \\ u(x,y)|_{\partial\Omega} &= 1 + xy. \end{aligned}$$

この方程式を 5 点中心差分近似を用いて離散化し，真の解が $u(x,y) = 1 + xy$ となるように右辺 $G(x,y)$ を定めて数値実験を行った．ここで，格子点間の距離を $h = 1/513$ にした．得られた連立 1 次方程式の次元は $262,144$ となる．また，$Dh = 2^{-3}, 2^{-4}, 2^{-5}, 2^{-6}, 2^{-7}$ の 5 通りで数値実験を行った．数値例 1 の結果を表 3.1 に示す．各 Dh の値において，算法ごとの計算時間が最も短かったものに下線を引いてある．表 3.1 の (数値) は，最大反復回数で収束条件を満たさなかった場合の残差ノルムの常用対数値である．

表 3.1 によると，GMRES(10) 法，GMRES(20) 法と GMRES(30) 法は，全ての Dh に対して最大反復回数で収束しなかった．GMRES(40) 法は $Dh = 2^{-5}$ の時だけ収束した．GMRES(50) 法は $Dh = 2^{-5}, 2^{-6}, 2^{-7}$ の時に収束した．これに対して，RITZ-GMRES 法は全てのケースで収束した．しかも，処理時間は GMRES(m) 法の収束した場合と比べておよそ半分以下である．

GMRES 法では，アーノルディ過程の直交化処理に，最も計算時間を必要とする．その処理時間はクリロフ部分空間の次元に依存する．本論文の数値実験では，RITZ-GMRES 法

表 3.1 数値例 1 の結果 （T: 計算時間 (秒)，I: 反復回数）

Dh	2^{-3}		2^{-4}		2^{-5}		2^{-6}		2^{-7}	
Method	I	T	I	T	I	T	I	T	I	T
GMRES(10)	(-5.19)	2039	(-5.35)	2035	(-5.51)	2042	(-5.44)	2041	(-4.85)	2037
GMRES(20)	(-6.35)	3090	(-6.66)	3084	(-7.60)	3091	(-7.68)	3081	(-6.50)	3083
GMRES(30)	(-8.66)	4138	(-9.63)	4133	(-9.08)	4135	(-10.11)	4126	(-8.47)	4131
GMRES(40)	(-10.52)	5189	(-11.49)	5190	18235	4741	(-11.30)	5189	(-11.09)	5185
GMRES(50)	(-11.98)	6252	(-10.06)	6243	11911	3711	14709	4592	19340	6030
RITZ-GMRES	19305	1976	19225	2693	18693	1187	10170	1187	9267	1163

(数値) は，最大反復回数で収束条件を満たさなかった場合の残差ノルムの常用対数値である

表 3.2 数値例 1: RITZ-GMRES 法のリスタート周期 m

Dh	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
m の平均値	4.77	4.92	5.02	5.85	5.65
m の最大値	25	27	29	31	28

の最大リスタート周期（つまりクリロフ部分空間の最大次元）を 50 にしたが，実際のリスタート周期の最大値と平均値を表 3.2 で示す．

表 3.2 から RITZ-GMRES 法の 1 回の反復に必要な処理時間が GMRES(31) 法より少ないことがわかる．リスタート周期の平均値は 6 以下である．ここで，$Dh = 2^{-4}$ を取り上げで，RITZ-GMRES 法の実際のリスタート周期を図 3.2 で示す．この場合，実際の最大リスタート周期は 27，平均値は約 5 なので，RITZ-GMRES 法と GMRES(5) 法，GMRES(27) 法との比較を行った．RITZ-GMRES 法と GMRES(5) 法，GMRES(27) 法の反復回数に対する残差ノルムの収束の様子を図 3.3 で示し，処理時間に対する残差ノルムの収束の様子を図 3.4 で示す．図 3.3 と図 3.4 から，RITZ-GMRES 法は GMRES(5) 法や GMRES(27) 法と比べて，収束が改善されていることがわかる．これらの結果は第 3.2 節で述べた事柄を裏付け，最大 Ritz 値と最大調和 Ritz 値を考慮したリスタート手法の有効性を示している．

ここで，Ritz 値と調和 Ritz 値の差と残差ノルムの収束との関連性を示すために，$Dh = 2^{-4}$ における 6897 回目の反復を取り上げて，Ritz 値と調和 Ritz 値の差について考察する．6897 回目の反復前後の最大 Ritz 値と最大調和 Ritz 値の差を図 3.5 で示す．x 軸は反復回数，y 軸は最大 Ritz 値と最大調和 Ritz 値の差である．ただし，y 軸の表示範囲を [0，0.1] にした．図 3.5 から，6890 回目反復から 6898 回目の反復にかけて，RITZ-GMRES 法の最大 Ritz 値と最大調和 Ritz 値の差が GMRES(5) 法や GMRES(27) 法より小さいことがわかる．この時，図 3.3 から RITZ-GMRES 法の残差ノルムの収束は GMRES(5) 法や GMRES(27) 法の

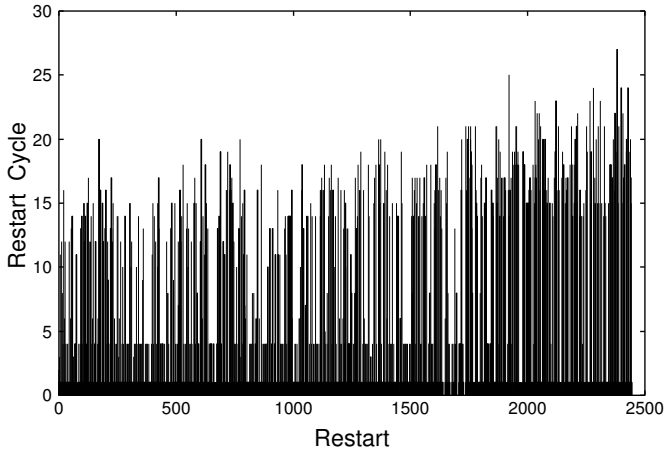


図 3.2 数値例 1: RITZ-GMRES 法のリスタート周期 ($Dh = 2^{-4}$)

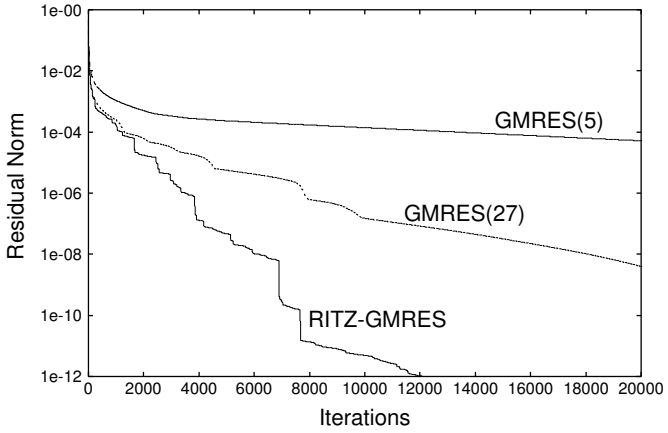


図 3.3 数値例 1: 残差ノルム vs. 反復回数 ($Dh = 2^{-4}$)

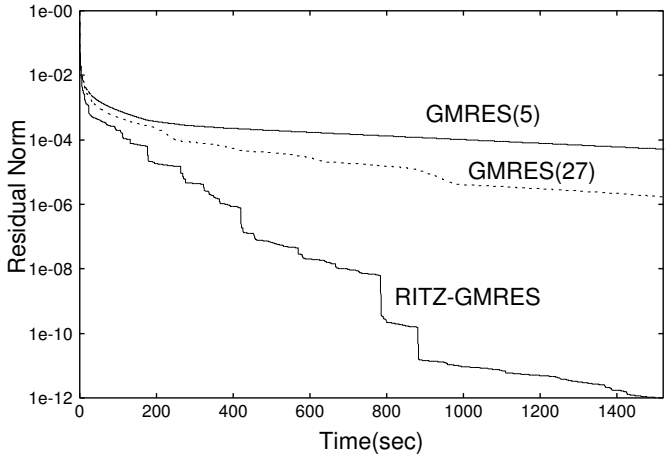


図 3.4 数値例 1: 残差ノルム vs. 計算時間 ($Dh = 2^{-4}$)

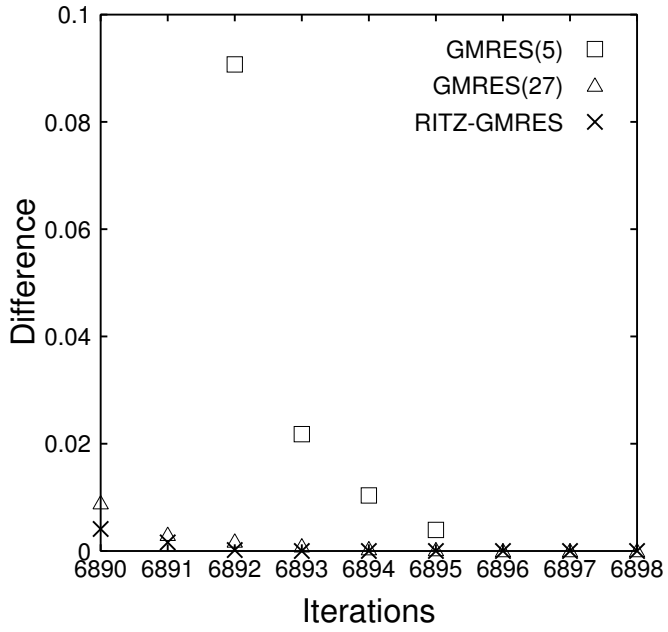


図 3.5 数値例 1: RITZ-GMRES 法の $|\mu_{\max} - \bar{\mu}_{\max}|$ の分布 ($Dh = 2^{-4}$)

表 3.3 数値例 2 の結果 （T:計算時間 (秒)， I：反復回数）

Method	I	T
GMRES(10)	—	—
GMRES(20)	7868	84
GMRES(30)	7588	102
GMRES(40)	5614	92
GMRES(50)	3187	61
RITZ-GMRES	5951	36

—: 最大反復回数で収束しなかった場合

残差ノルムの収束より優れていることがわかる．これは第 3.2 節で述べた事柄と一致する．即ち，Ritz 値と調和 Ritz 値の差が小さい時，GMRES 法の収束が速い．

また，図 3.5 から，6890 回目反復から 6898 回目の反復にかけて，RITZ-GMRES 法の最大 Ritz 値と最大調和 Ritz 値の差が減少し続けることがわかる．それに対して，RITZ-GMRES 法の残差ノルムも順調に収束していることが図 3.3 から確認できる．特に，6897 回目の反復において 1 回の反復だけで，RITZ-GMRES 法の残差ノルムの常用対数値は -8.39 から -8.71 まで大幅に減少している．これも 3.2 節で述べた事柄と一致する．即ち，Ritz 値と調和 Ritz 値の差が小さくなる時，GMRES 法の収束も良くなる．

[数値例 2] Matrix Market[28] において提供されている実非対称正方行列 MEMPLUS を考える．MEMPLUS は次元 17758，126150 個の非ゼロ要素を持つ疎な実行列である．行列

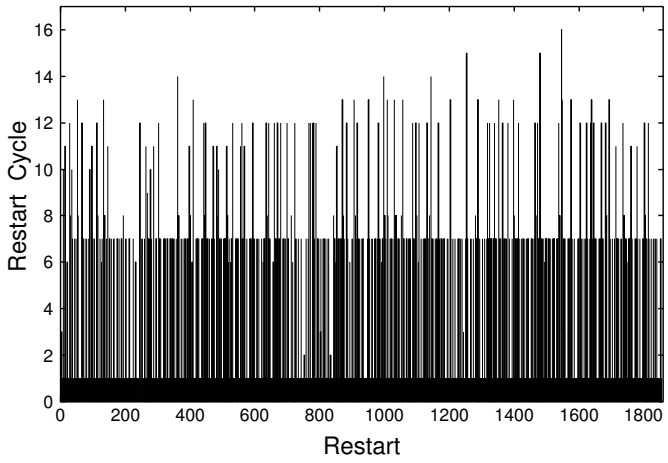


図 3.6 数値例 2: RITZ-GMRES 法のリスタート周期

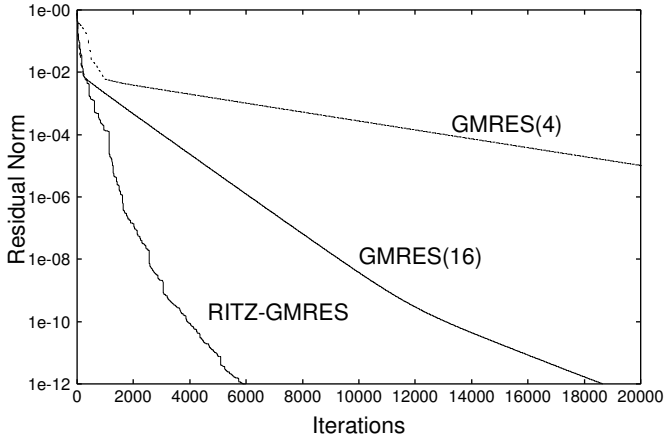


図 3.7 数値例 2: 残差ノルム vs. 反復回数

MEMPLUS を連立 1 次方程式の係数行列とする．連立 1 次方程式の右辺も Matrix Market において提供されているデータを使用する．

数値実験の結果を表 3.3 に示す．数値例 1 と同様に，計算時間が最も短かったものに，下線を引いてある．表 3.3 から，RITZ-GMRES 法の処理時間が最も短かったことがわかる．RITZ-GMRES 法の実際のリスタート周期の最大値と平均値は 16 と 3.21 である．つまり，RITZ-GMRES 法の一回の反復において必要な記憶容量と処理時間が GMRES(16) 法よりも少ない．RITZ-GMRES 法の実際のリスタート周期を図 3.6 で示す．

RITZ-GMRES 法と GMRES(4) 法，GMRES(16) 法の反復回数に対する残差ノルムの収束の様子を図 3.7 に示し，処理時間に対する残差ノルムの収束の様子を図 3.8 で示す．図 3.7 と図 3.8 から RITZ-GMRES 法が良い収束をしていることは明らかである．

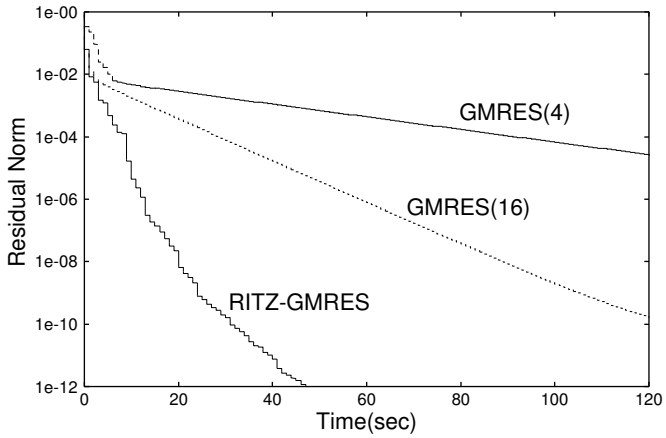


図 3.8 数値例 2: 残差ノルム vs. 計算時間

最後に，最大 Ritz 値と最大調和 Ritz 値の差と残差ノルムの収束との関連性を示す．2257 回目の反復付近の各反復の最大 Ritz 値と最大調和 Ritz 値の差を図 3.9 で示す．x 軸は反復回数，y 軸は最大 Ritz 値と最大調和 Ritz 値の差である．ただし，y 軸の表示範囲を [0，0.1] にした．図 3.9 から，2249 回目の反復から 2257 回目の反復にかけて，RITZ-GMRES 法の最大 Ritz 値と最大調和 Ritz 値の差は GMRES(4) 法や GMRES(16) 法より小さいことがわかる．この時，RITZ-GMRES 法の残差ノルムの収束は，GMRES(4) 法や GMRES(16) 法より優れていることが図 3.7 から確認できる．これは，最大 Ritz 値と最大調和 Ritz 値の差が小さい時，GMRES 法の収束が速いことを示している．

また，図 3.9 から，2249 回目の反復から 2257 回目の反復にかけて，RITZ-GMRES 法の最大 Ritz 値と最大調和 Ritz 値の差が減少し続けることもわかる．これに対して，RITZ-GMRES 法の残差ノルムも順調に収束していることが図 3.7 から確認できる．特に，2257 回目の反復において，RITZ-GMRES 法の残差ノルムの常用対数値は -7.77 から -7.94 まで大幅に減少している．これは，最大 Ritz 値と最大調和 Ritz 値の差が小さくなる時，GMRES 法の残差ノルムの収束が良くなることを示している．

3.4 本章のまとめ

GMRES(m) 法は非対称で，正則な大型疎行列を係数として持つ連立 1 次方程式を解くための有効な算法の 1 つである．しかし，その収束はリスタート周期 m に大きく依存する．最適な m を決めるのは難しいことである．

本論文では，係数行列 A の近似固有値である Ritz 値と調和 Ritz 値に注目し，適応的なリスタート手法を提案した．GMRES 法において，Ritz 値と調和 Ritz 値は低いコストで求め

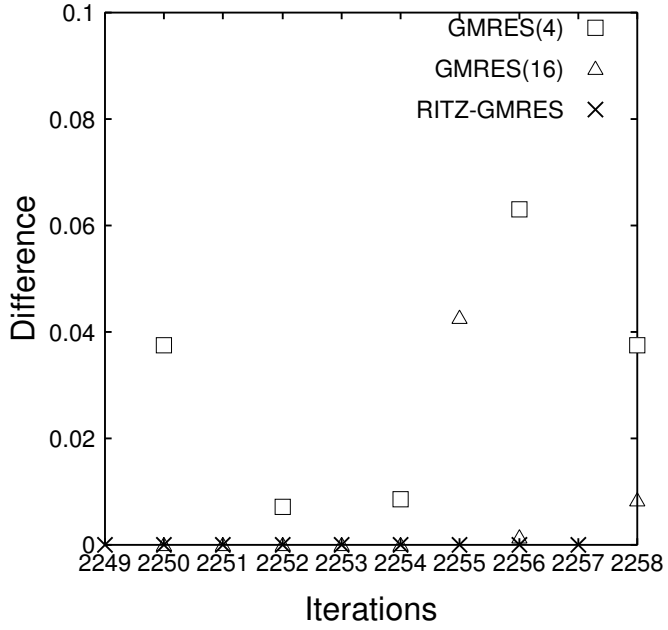


図 3.9 数値例 2: RITZ-GMRES 法の $|\mu_{\max} - \bar{\mu}_{\max}|$ の分布

られる．しかも GMRES 法の収束はこれら 2 つの値から評価できる．すなわち，Ritz 値と調和 Ritz 値の差が大きければ，GMRES 法の残差ノルムの収束は遅くなる．これに対して，GMRES 法の残差ノルムの収束が早ければ，Ritz 値と調和 Ritz 値の差が小さくなる．本手法は，GMRES(m) 法の中で簡単に求められる Ritz 値と調和 Ritz 値を利用して，リスタートするタイミングを決定している．このリスタート手法の有効性を示すため，固定したリスタート周期を持つ古典的な GMRES(m) 法との比較を中心に，数値実験を行った．楕円型偏微分方程式の境界値問題の離散化から得られる方程式の数値実験によると，GMRES(m) 法と比べて，提案したリスタート手法を利用した GMRES(m) 法には，次のような利点がある．

- リスタート周期 m を設定するユーザの負担を軽減できる
- 不必要な直交処理を削減でき，早い収束が得られる可能性がある
- 実装が簡単である

また，このリスタート手法は前処理付きの GMRES(m) 法にも適用できる．DEFL-ATED-GMRES(m, k) 法に，提案したリスタート手法を実装し，その有効性を確認した．その詳細については，文献 [50] を参照のこと．

第 4 章

近似逆行列による行列の前処理

第 1 章の 1.1 節で述べたように，式 (1.1) に，行列の前処理を行うことによって，近似解が求まるまでの反復回数を減少させることができる．前処理には，右前処理と左前処理がある．右前処理とは，式 (1.1) の係数行列 A の右側から前処理行列 M を掛けて

$$\begin{cases} AM\boldsymbol{y} = \boldsymbol{b} \\ \boldsymbol{x} = M\boldsymbol{y} \end{cases}$$

のように係数行列の変換をして近似解を計算する手法である．左前処理は係数行列 A の左側から前処理行列を掛ける手法である．

$$MA\boldsymbol{x} = M\boldsymbol{b}$$

また，両側から掛ける前処理もある．右前処理は，残差ベクトルの値を変えないため，良く使われる．本論文でも，右前処理を利用する．

前処理行列 M は，AM が A より条件数が小さいものを選ぶ．一般的には，AM が単位行列 I_n に近いほうが望ましい．従って，A の近似逆行列を前処理行列にする手法が良く使われる．また，AM の固有値が原点から離れた所に局在する場合にも良い収束が得られる．このように少ない条件数と良い固有値分布のほか，前処理行列 M が簡単に求められることと低いコストで実装できることも必要条件の 1 つである．

前処理行列 M の選択について，現在では様々な前処理が提案されている．本章では，現在主流となっている ILU 分解，MR 法と最近提案されている Sherman-Morrison 公式を用いて，近似逆行列を計算する AISM 法について述べる．

4.1 ILU 分解による行列の前処理とその並列化

4.1.1 ILU 分解による行列の前処理

ILU 分解は不完全 LU 分解のことである [10]．LU 分解とは，行列 A を下三角行列 L と上三角行列 U の積，すなわち

$$A = LU$$

と分解することを言う．しかし，例え行列 A が疎な行列であっても，行列 L と U は疎な行列とは限らない．実際，フィルイン現象 (0 であったところに，0 でなくなる現象) が起きるため，L と U は密行列になる場合が多い．このため，計算量が多くなり，余計なメモリ容量が要求されることになる．L と U の疎な性質を保つために，係数行列 A の非零要素の位置から左右それぞれ k 個目までのフィルインのみを許すような不完全 LU 分解 (ILU 分解) のことを ILU(k) 法と表記する．この場合，

$$A \approx LU$$

となる．ただし，ここで L は疎な下三角行列，U は疎な上三角行列である．

行列の ILU 分解の逆行列 $(LU)^{-1}$ を前処理行列 M にする前処理技法が良く使われている．ただし，$(LU)^{-1}$ は明示的に計算，保存されない．この前処理行列 $M = (LU)^{-1}$ を反復法に適用するときは，$\boldsymbol{w} = (LU)^{-1}\boldsymbol{v}$ を計算する必要がある．$(LU)^{-1}$ は明示的に計算，保存されないため，行列とベクトルの積として計算できない．この場合，連立 1 次方程式 $LU\boldsymbol{w} = \boldsymbol{v}$ を解くことによって，$\boldsymbol{w}$ を求めることになる．$\boldsymbol{w}$ の計算は以下のような行われる．

$$L\tilde{\boldsymbol{v}} = \boldsymbol{v} \tag{4.1}$$

$$U\boldsymbol{w} = \tilde{\boldsymbol{v}} \tag{4.2}$$

$\tilde{\boldsymbol{v}}$ はガウス消去法の前進消去によって計算され，$\boldsymbol{w}$ は後退代入によって計算する．行列 L と U が疎な行列のため，$\boldsymbol{w}$ は低いコストで求めることができる．

4.1.2 ILU 分解による行列の前処理の並列化

すでに述べたように，ILU 分解による行列の前処理を反復法に適用するときの前処理行列とベクトルの積はガウス消去法の前進消去と後退代入によって計算する．行列 L と U が疎な行列のため，計算コストは低い．しかし，ガウス消去法の前進消去と後退代入には，依存関係がある．例えば，ガウス消去法の前進消去で $\tilde{\boldsymbol{v}}$ を次式で計算する．

$$\tilde{\boldsymbol{v}}_i = \frac{1}{L_{i,i}}(\boldsymbol{v}_i - \sum_{j=1}^{i-1} L_{i,j}\tilde{\boldsymbol{v}}_j)$$

しかし，$\tilde{\boldsymbol{v}}_i$ を計算するために，$\tilde{\boldsymbol{v}}_j,\ (j = 1, \ldots, i-1)$ が必要である．つまり，$\tilde{\boldsymbol{v}}_j,\ (j = 1, \ldots, i-1)$ の計算が終わらないと，$\tilde{\boldsymbol{v}}_i$ の計算を始められない．同様に，後退代入で $\boldsymbol{w}_i$ を計算するために，$\boldsymbol{w}_j,\ (j = j+1, \ldots, n)$ を必要とするので，$\boldsymbol{w}_j,\ (j = j+1, \ldots, n)$ の計算が終わらないと，$\boldsymbol{w}_i$ の計算を始められない．このため，ILU 分解による行列の前処理を並列化することに難点があり，並列処理を行うにはさらに工夫が必要である．

ここで，森屋ら [33] の ILU 分解による行列の前処理の並列実装について簡単に述べる．この並列計算では，行列 $L,\ U$ の行ベクトルは均等に m 行ずつのブロック分割される．説明のため，行列の次元 n は m で割り切れるものとする．ベクトル $\boldsymbol{v}$，$\tilde{\boldsymbol{v}}$，$\boldsymbol{w}$ も同様に分割され，l 番目の PE が l 番目のブロックを担当する．各 PE は自分の担当する $\tilde{\boldsymbol{v}}$，$\boldsymbol{w}$ の l 番目のブロックだけ計算する．ここで，式 (4.1) をガウス消去法の前進消去で $\tilde{\boldsymbol{v}}$ を解く場合を取り上げて，並列実装について説明する．l 番目の PE が担当する $\tilde{\boldsymbol{v}}$ の要素 $\tilde{\boldsymbol{v}}_i$ の計算式を次式のように変形する．

$$\begin{aligned} \tilde{\boldsymbol{v}}_i \quad = \quad & \frac{1}{L_{i,i}}(\boldsymbol{v}_i - \sum_{j=1}^{(l-1)m} L_{i,j}\tilde{\boldsymbol{v}}_j - \sum_{j=(l-1)m+1}^{i-1} L_{i,j}\tilde{\boldsymbol{v}}_j), \\ & i = (l-1)m+1, \ldots, lm \end{aligned} \tag{4.3}$$

この式の右辺の () の中の第 2 項を計算するために，$\tilde{v}_j,\ (j = 1, \ldots, (l-1)m)$ を別の PE から受信したものを使う．計算と通信を交互に行うことによって，部分的な並列計算が可能となる．しかし，各 PE が自分の担当する $\boldsymbol{v}_j$ をすべて計算してから，別の PE に送信すると，ほかの PE の待ち時間が長くなるので，各 PE が自分の担当する $\tilde{\boldsymbol{v}}_j$ を $\tilde{m}(\leq m)$ ずつのみを計算して，その結果をほかの PE に送信することも可能である．例えば，$\tilde{m}$ を 2 とすると，最初に 1 番目の PE が $\tilde{\boldsymbol{v}}_1, \tilde{\boldsymbol{v}}_2$ を計算して，計算した結果を別の PE に通信する．次の段階で 2 番目以降の PE は $\tilde{\boldsymbol{v}}_1, \tilde{\boldsymbol{v}}_2$ を使って，担当する $\tilde{\boldsymbol{v}}_i$ を更新する．同時に PE1 は $\tilde{\boldsymbol{v}}_3,\ \tilde{\boldsymbol{v}}_4$ の計算をすることができる．

4.2 MR 法

MR 法 (最小残差法) は，Grote ら [19] と Huckel [23] によって提案された近似逆行列を計算する方法であり，反復解法を利用して前処理行列を計算する代表的な算法の 1 つである．

ここで，ベクトル $\boldsymbol{e}_j,\ \boldsymbol{m}_j$ をそれぞれ単位行列 I, 近似逆行列 M の j 番目の列ベクトルであるとする．MR 法は最急降下法を用いて

$$\boldsymbol{m}_j^{(k)} = \boldsymbol{m}_j^{(k-1)} + \alpha_j^{(k)}\boldsymbol{r}_j^{(k)}, \quad j = 1,\ 2,\ \ldots,\ n \tag{4.4}$$

となるような漸化式によって近似逆行列 M の j 番目の列ベクトルを計算する．ただし，$\boldsymbol{m}_j^{(k)}$ は最急降下法の k 回目の反復における $\boldsymbol{m}_j$ を表し，$\boldsymbol{r}_j^{(k)} = \boldsymbol{e}_j - A\boldsymbol{m}_j^{(k)}$ である．$\alpha_j^{(k)}$ は各ステップにおいて，最小 2 乗問題

$$\min_{m_j^{(k)}} \|\boldsymbol{e}_j - A\boldsymbol{m}_j^{(k)}\|_2^2, \quad j = 1,\ 2, \ldots, n \tag{4.5}$$

を解くことに求める．

MR 法においては，近似逆行列 M の列ベクトルを全く独立に考えるので，n 個の漸化式 (4.4) が各 PE に均等に割り当られ，他の PE と全く独立に計算できるので，並列化に適している．また，MR 法で計算された近似逆行列を前処理行列として，反復法に適用するとき，前処理行列とベクトルの積も普通の行列とベクトル積に帰着するので，この部分の並列実装もしやすいものである．

ILU 分解と比べて，MR 法の計算コストは高い．計算量を抑えるための工夫がいくつか提案されている [23, 24, 37]．$\boldsymbol{m}_j$ の非零要素に関しては，その絶対値がある閾値より小さい場合は切り捨てることによって，M の疎な性質を保つ．それから，MR 法の内部反復，つまり最急降下法の反復を一定の回数 imax で打ち切るように設定することになる．

4.3 Sherman-Morrison公式による近似逆行列法 (AISM法)

Bru ら [8] によって提案された AISM 法は，次の Sherman-Morrison 公式を利用して，行列の近似逆行列を求める方法である．

[Sherman-Morrison 公式] 与えられた正則行列 $B \in \boldsymbol{R}^{n\times n}$ とベクトル $\boldsymbol{x} \in \boldsymbol{R}^n$, $\boldsymbol{y} \in \boldsymbol{R}^n$ に対して，もし $r = 1 + \boldsymbol{y}^T B^{-1}\boldsymbol{x} \neq 0$ なら，行列 $A = B + \boldsymbol{x}\boldsymbol{y}^T$ も正則行列であり，その逆行列は

$$A^{-1} = B^{-1} - r^{-1}B^{-1}\boldsymbol{x}\boldsymbol{y}^T B^{-1} \tag{4.6}$$

となる．□

ここで，ベクトル $\{\boldsymbol{x}_k\}_{k=1}^n \in \boldsymbol{R}^n$，$\{\boldsymbol{y}_k\}_{k=1}^n \in \boldsymbol{R}^n$ と行列 $A_0 \in \boldsymbol{R}^{n\times n}$ は，次式を満たすものとする．

$$A = A_0 + \sum_{k=1}^{n} \boldsymbol{x}_k \boldsymbol{y}_k^T \tag{4.7}$$

式 (4.7) において $A_k = A_0 + \sum_{i=1}^{k} \boldsymbol{x}_i \boldsymbol{y}_i^T$ と置くと，$A_{k+1} = A_k + \boldsymbol{x}_{k+1}\boldsymbol{y}_{k+1}^T$，$A_n = A$ となる．もし $\boldsymbol{x}_k, \boldsymbol{y}_k$ と A_k が Sherman-Morrison 公式の条件を満たすならば，式 (4.6) を n 回適用すると，係数行列 A の逆行列は次式で計算できることになる．AISM 法はこれに基づい

て，近似逆行列を計算する．

$$A^{-1} = A_n^{-1} = A_0^{-1} - \sum_{k=1}^{n} r_k^{-1} A_{k-1}^{-1} \boldsymbol{x}_k \boldsymbol{y}_k^T A_{k-1}^{-1}$$

ただし，$r_k = 1 + \boldsymbol{y}_k^T A_{k-1}^{-1} \boldsymbol{x}_k$ である．上記式を整理すると

$$A_0^{-1} - A^{-1} = \sum_{k=1}^{n} r_k^{-1} A_{k-1}^{-1} \boldsymbol{x}_k \boldsymbol{y}_k^T A_{k-1}^{-1} \tag{4.8}$$

となる．ここで，式 (4.8) の右辺を行列形式に書き換えると，次式のようになる．

$$A_0^{-1} - A^{-1} = \Phi \Omega^{-1} \Psi^T \tag{4.9}$$

ただし，$\Phi = [A_0^{-1}\boldsymbol{x}_1,\, A_1^{-1}\boldsymbol{x}_2,\, \ldots,\, A_{n-1}^{-1}\boldsymbol{x}_n]$，$\Omega^{-1} = \mathrm{diag}[r_1^{-1},\, r_2^{-1},\, \ldots,\, r_n^{-1}]$，$\Psi = [\boldsymbol{y}_1^T A_0^{-1},\, \boldsymbol{y}_2^T A_1^{-1},\, \ldots,\, \boldsymbol{y}_n^T A_{n-1}^{-1}]$ である．行列 Φ と Ψ を計算するには，A_k^{-1} を計算し，保存する必要がある．しかし，新たなベクトル $\boldsymbol{u}_k,\, \boldsymbol{v}_k$ を

$$\boldsymbol{u}_k := \boldsymbol{x}_k - \sum_{i=1}^{k-1} \frac{\boldsymbol{v}_i^T A_0^{-1} \boldsymbol{x}_k}{r_i} \boldsymbol{u}_i, \quad \boldsymbol{v}_k := \boldsymbol{y}_k - \sum_{i=1}^{k-1} \frac{\boldsymbol{y}_k^T A_0^{-1} \boldsymbol{u}_i}{r_i} \boldsymbol{v}_i$$

というように定義すれば，次式が成立する．

$$A_{k-1}^{-1} \boldsymbol{x}_k = A_0^{-1} \boldsymbol{u}_k, \tag{4.10}$$

$$\boldsymbol{y}_k^T A_{k-1}^{-1} = \boldsymbol{v}_k^T A_0^{-1}, \tag{4.11}$$

$$r_k = 1 + \boldsymbol{y}_k^T A_0^{-1} \boldsymbol{u}_k = 1 + \boldsymbol{v}_k^T A_0^{-1} \boldsymbol{x}_k. \tag{4.12}$$

ここで，式 (4.10) と式 (4.11) を式 (4.9) に代入すると

$$A_0^{-1} - A^{-1} = A_0^{-1} U \Omega^{-1} V^T A_0^{-1} \tag{4.13}$$

となる．ただし，$U = [\boldsymbol{u}_1,\, \boldsymbol{u}_2,\, \ldots,\, \boldsymbol{u}_n]$，$V = [\boldsymbol{v}_1,\, \boldsymbol{v}_2,\, \ldots,\, \boldsymbol{v}_n]$ である．

さらに，Bru ら [8] は，A_0，$\boldsymbol{x}_k$，$\boldsymbol{y}_k$ について，次のような選択を提案している．

$$A_0 = sI_n, \quad (s > 0), \quad \boldsymbol{x}_k = \boldsymbol{e}_k, \quad \boldsymbol{y}_k = (\boldsymbol{a}^k - \boldsymbol{a}_0^k)^T.$$

ただし，$I_n \in \boldsymbol{R}^{n \times n}$ は単位行列であり，$\boldsymbol{e}_k \in \boldsymbol{R}^n$ は I_n の k 番目の列ベクトルである．$\boldsymbol{a}^k$ と $\boldsymbol{a}_0^k$ は行列 A と A_0 の k 番目の行ベクトルである．ここで，A_0，$\boldsymbol{x}_k$，$\boldsymbol{y}_k$ を式 (4.13) に代入すると，次式が得られることになる．

$$A^{-1} = s^{-1} I_n - s^{-2} U \Omega^{-1} V^T \tag{4.14}$$

```
for k = 1, ..., n do
    x_k = e_k,  y_k = (a^k − s e_k)^T,
    u_k = x_k, v_k = y_k
    for i = 1, ..., k − 1
        u_k = u_k − ((v_i)_k / s r_i) u_i
        v_k = v_k − (y_k^T u_i / s r_i) v_i
    end for
    for i = 1, ..., n
        if |(u_k)_i| < tolU dropoff (u_k)_i
        if |(v_k)_i| < tolV dropoff (v_k)_i
    end for
    r_k = 1 + (v_k)_k / s
end for
```

図 4.1 AISM の算法

また，$\boldsymbol{u}_k$，$\boldsymbol{v}_k$ は，次式のようになる．

$$\boldsymbol{u}_k = \boldsymbol{x}_k - \sum_{i=1}^{k-1} \frac{(\boldsymbol{v}_i)_k}{sr_i}\boldsymbol{u}_i, \quad \boldsymbol{v}_k = \boldsymbol{y}_k - \sum_{i=1}^{k-1} \frac{\boldsymbol{y}_k^T\boldsymbol{u}_i}{sr_i}\boldsymbol{v}_i \tag{4.15}$$

このように A_0，$\boldsymbol{x}_k$，$\boldsymbol{y}_k$ を選べば，A_0^{-1} と $\boldsymbol{u}_k$，$\boldsymbol{v}_k$ が簡単に求められるだけでなく，行列 U の正則性も保証できることになる．さらに，$s \notin \sigma(A)$ なら，行列 V も正則である．ただし，$\sigma(A)$ は行列 A のスペクトル半径である．なお，これらの詳細に関しては，Bru ら [8] を参照して欲しい．また，U と V の疎な性質を維持するために，非ゼロ要素の切り捨ての処理を行う必要がある．従って，次式のようになる．

$$A^{-1} \approx s^{-1}I_n - s^{-2}U\Omega^{-1}V^T. \tag{4.16}$$

ここで，AISM 法の算法をまとめると，図 4.1 のようになる．

式 (4.15) から，$\boldsymbol{u}_k$ と $\boldsymbol{v}_k$ を計算するために，$\boldsymbol{v}_i$，$\boldsymbol{u}_i$，r_i $(i = 1, \ldots, k-1)$ を必要とする．すなわち，ベクトルどうしに依存関係があるので，並列計算は難しい．このため，AISM 法は並列計算に不向きであるが，算法の部分的な並列化は可能である．

第 5 章

AISM法の部分並列化

さまざまな数値実験によれば，MR 法と比べて，AISM 法は多くの場合，前処理としての性能が安定しているといわれている [34]．しかし，係数行列の次元が大きい場合，その近似逆行列を計算するために，膨大な時間を要するケースがしばしばある．しかも，計算過程においてベクトルどうしの依存関係が存在しているため，並列化には不向きであることが指摘されている．このような難点を改善するために，本章では Naik [36] の提案した実装法を利用して前処理行列の計算を部分的に並列化することを考える．まず，第 5.1 節で Naik の並列手法について述べる．次に，Naik の 1 次元分割の実装を用いて，AISM 法の部分並列化を提案する．第 5.3 節の数値実験を通して，AISM 法の前処理法としての性能と提案する部分並列化の有効性について考察する．最後に，第 5.4 節で，本章のまとめについて述べる．

5.1 Naikの並列手法

Naik [36] の論文では，分散メモリ型の並列計算環境において，Benchmark BT という非優位対角な 5×5 のブロックを持つ 3 重ブロック対角行列を係数とする線形システムを LU 分解で解く過程がパイプライン制御により部分的に並列化されている．線形システムの係数行列は，3 次元の偏微分方程式の境界値問題から有限差分法によって得られたものである．

分散メモリ型の並列計算環境におけるデータの各 PE への割当てについて，Naik は 3 つの方法を考えた．それらは 1 次元分割 (1D unipartitioning)，2 次元分割 (2D unipartitioning) と 3 次元分割 (3D unipartitioning) である．これらの分割で領域をそれぞれ 1 方向，2 方向，3 方向に分割することになる．これら 3 通りの方法により分割された領域の各格子点上における解を LU 分解によって求める．本論文では，1 次元分割だけを用いて，AISM 法の部分並列化を行うので，ここでは，1 次元分割だけについて，簡単に述べる．

1 次元分割とは，領域を図 5.1 のように 1 方向のみに分割する方法である．分割された部

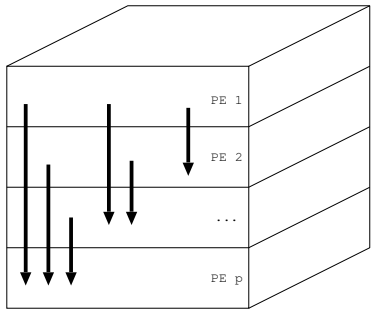


図 5.1 1 次元分割

分領域の数を p とすると，i 番目の領域に属する格子点の解は i 番目の PE(PEi) で解くことになる．この図における矢印は依存関係を示している．ある PE に属する格子点の情報は，自分より上の領域に属する格子点の情報を受信することではじめて求めることができる．例えば，PE2 に属する格子点における解を求めるために，PE1 に属する格子点の解を受信しないと，始めることができない．同様に，PEp に属する格子点における解を求めるために，PEi，$(i = 1, \ldots, p-1)$ に属する格子点の解を受信する必要がある．

5.2 AISM 法の部分並列化

本節では，前節で述べた Naik [36] の 1 次元分割法を用いて，AISM 法の部分並列化を提案する．まず，第 5.2.1 小節で，データの各 PE への割当てについて述べ，その次に，第 5.2.2 小節で，AISM 法の部分並列計算について述べる．

5.2.1 データの割当て

AISM 法で近似逆行列を計算する際に現れる行列を Naik が考えている 1 次元分割法を利用して，1 方向にのみ分割する．分割する方向について，AISM 法では，U と V は列ベクトルごとに計算されるので，列ベクトルを分割せず，横方向での分割を行う．従って，行列 U，V，Ω の列ベクトルを図 5.2 のように，横へ 1 方向のみに各 PE に均等に分散させる．図 5.1 と同様に，図における矢印は依存関係を示している．つまり，各列ベクトルを計算する際に，自分よりも左の PE に属する列ベクトルの情報を利用するので，通信を行う必要がある．

具体的に，行列 U,V,Ω の列ベクトルを図 5.3 のように，各 PE に分割する．i 番目の PE

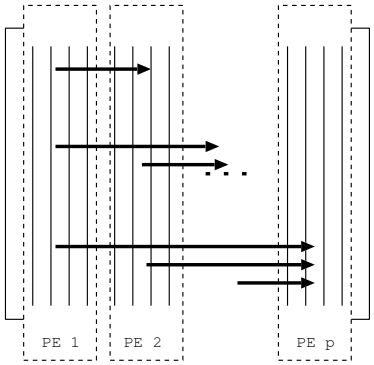


図 5.2 行列の 1 次元分割

$$
\begin{aligned}
U &= \{\underbrace{\boldsymbol{u}_{s[1]},\ldots,\boldsymbol{u}_{e[1]}}_{\mathrm{PE1}},\underbrace{\boldsymbol{u}_{s[2]},\ldots,\boldsymbol{u}_{e[2]}}_{\mathrm{PE2}},\ldots\ldots,\underbrace{\boldsymbol{u}_{s[p]},\ldots,\boldsymbol{u}_{e[p]}}_{\mathrm{PE}p}\} \\
V &= \{\underbrace{\boldsymbol{v}_{s[1]},\ldots,\boldsymbol{v}_{e[1]}}_{\mathrm{PE1}},\underbrace{\boldsymbol{v}_{s[2]},\ldots,\boldsymbol{v}_{e[2]}}_{\mathrm{PE2}},\ldots\ldots,\underbrace{\boldsymbol{v}_{s[p]},\ldots,\boldsymbol{v}_{e[p]}}_{\mathrm{PE}p}\} \\
\Omega &= \{\underbrace{r_{s[1]},\ldots,r_{e[1]}}_{\mathrm{PE1}},\underbrace{r_{s[2]},\ldots,r_{e[2]}}_{\mathrm{PE2}},\ldots\ldots,\underbrace{r_{s[p]},\ldots,r_{e[p]}}_{\mathrm{PE}p}\}
\end{aligned}
$$

図 5.3 行列 U, V, Ω の PE への割当て

が $s[i]$ から $e[i]$ までのベクトルの計算を担当する．また，行列 Ω は対角行列であるため，対角要素のみを各 PE に分割する．

配列 $s[i]$,$e[i]$,$(i = 1, \ldots, p)$ の値について，各 PE に均等に分割するように設定する．本論文では，図 5.4 のように設定する．すなわち，行列 A の次元 n が p で割り切れる場合，ベクトルを各 PE に均等に分割する．n が p で割り切れない場合，最初の $n \bmod p$ 個の PE は 1 個ずつ多めに分担する．ただし，n，p は行列 A の次元数と使用する PE 台数である．例えば行列 A の次元数 $n = 128$，使用する PE 台数 $p = 6$ の場合，PE1 と PE2 の担当する行列 U の列ベクトル数は 22 になり，それ以外の PE が 21 個の列ベクトルを担当することになる．行列 V と Ω の分割は U と同じである．

5.2.2 AISM 法の部分並列計算

ここでは，1 次元分割を用いた AISM 法の部分並列計算について述べる．まず，行列 U と V の列ベクトルの計算について考える．式 (4.15) から，U，V の列ベクトル $\boldsymbol{u}_k$，$\boldsymbol{v}_k$ を

```
1:  if n mod p = 0 then
1:    for i = 1, ..., p
1:      s[i] = (n/p) * (i − 1) + 1
1:      e[i] = (n/p) * i
1:    end for
1:  else
1:    for i = 1, ..., (n mod p)
1:      s[i] = (n/p + 1) * (i − 1) + 1
1:      e[i] = (n/p + 1) * i
1:    end for
1:    for i = (n mod p) + 1, ..., n
1:      s[i] = e[i − 1] + 1
1:      e[i] = s[i] + n/p − 1
1:    end for
1:  endif
```

図 5.4 配列 $s[i]$，$e[i]$ の設定 (PEi は $s[i]$ 番目から $e[i]$ 番目までの列ベクトルの計算を担当)

計算するために，$\boldsymbol{v}_i$，$\boldsymbol{u}_i$，r_i $(i = 1, \ldots, k-1)$ の情報が必要となる．ここで，$\boldsymbol{u}_k$ と $\boldsymbol{v}_k$ の計算に必要な情報を PE ごとに分けて，$\boldsymbol{u}_k$ と $\boldsymbol{v}_k$ の計算式 (4.15) を書き直すと，次式になる．

$$\boldsymbol{u}_k = \boldsymbol{x}_k \underbrace{- \sum_{i=s[1]}^{e[1]} \frac{(\boldsymbol{v}_i)_k}{sr_i}\boldsymbol{u}_i}_{\text{PE1 から通信}} \underbrace{- \sum_{i=s[2]}^{e[2]} \frac{(\boldsymbol{v}_i)_k}{sr_i}\boldsymbol{u}_i}_{\text{PE2 から通信}} \ldots \underbrace{- \sum_{i=s[l-1]}^{e[l-1]} \frac{(\boldsymbol{v}_i)_k}{sr_i}\boldsymbol{u}_i}_{\text{PE}(l-1)\text{ から通信}} \underbrace{- \sum_{i=s[l]}^{k-1} \frac{(\boldsymbol{q}_k, \boldsymbol{u}_i)}{sr_i}\boldsymbol{v}_i}_{\text{通信なしで計算可能}} \tag{5.1}$$

$$\boldsymbol{v}_k = \boldsymbol{y}_k \underbrace{- \sum_{i=s[1]}^{e[1]} \frac{(\boldsymbol{q}_k, \boldsymbol{u}_i)}{sr_i}\boldsymbol{v}_i}_{\text{PE1 から通信}} \underbrace{- \sum_{i=s[2]}^{e[2]} \frac{(\boldsymbol{q}_k, \boldsymbol{u}_i)}{sr_i}\boldsymbol{v}_i}_{\text{PE2 から通信}} \ldots \underbrace{- \sum_{i=s[l-1]}^{e[l-1]} \frac{(\boldsymbol{q}_k, \boldsymbol{u}_i)}{sr_i}\boldsymbol{v}_i}_{\text{PE}(l-1)\text{ から通信}} \underbrace{- \sum_{i=s[l]}^{k-1} \frac{(\boldsymbol{q}_k, \boldsymbol{u}_i)}{sr_i}\boldsymbol{v}_i}_{\text{通信なしで計算可能}}$$

ただし，$\boldsymbol{u}_k$ と $\boldsymbol{v}_k$ は l 番目の PE に所属するとする．式 (5.1) で示すように，PEl 以外の PE に所属するベクトルの情報を利用する際に，その PE からの通信を行う必要がある．同じ PEl に所属するベクトルの情報を利用する際は，PE 間の通信を行う必要はない．具体的に，並列計算を行なう際に，まず $\boldsymbol{u}_k$ と $\boldsymbol{v}_k$ に初期値を設定する．その後，PE1 から必要な情報を受信し，$\boldsymbol{u}_k$ と $\boldsymbol{v}_k$ を更新する．つまり，式 (5.1) の右辺 2 番目の部分を計算する．その次に，PE2 から必要な情報を受信し，式 (5.1) の右辺 3 番目の部分を計算する．このよ

うに，計算と通信を交互に行い，最後に，PE$(p-1)$ から必要な情報を受信し，$\boldsymbol{u}_k$ と $\boldsymbol{v}_k$ を更新する．ここで，受信する必要がなくなるので，$\boldsymbol{u}_k$ と $\boldsymbol{v}_k$ の式 (5.1) の右辺の最後の部分と，Ω の対角要素 $r_k = 1 + (\boldsymbol{v}_k)_k/s$ を計算する．本論文では，この部分の計算を最終計算と言う．

ここで，$p = 4$ を例にして，並列計算の詳細について述べる．便宜上，通信の最小単位であるブロック G_i を次式で定義する．

$$G_i := \{\boldsymbol{u}_i,\ \boldsymbol{v}_i,\ r_i\}$$

従って，l 番目の PE は $G_i, (s[l] \le i \le e[l])$ の計算と送信を担当する．G_k の計算には $G_1, \ldots, G_{k-1}$ が必要となる．部分並列計算は以下のように行われる．まず，PE1 では，ブロック $G_i, (s[1] \le i \le e[1])$ を計算する．計算が終わったら，$G_i, (s[1] \le i \le e[1])$ を PE2，PE3，PE4 に送信する．次に，PE2，PE3，PE4 では，PE1 から受信したブロック $G_i, (s[1] \le i \le e[1])$ を使って，自分の担当するブロックを更新する．この段階で，PE2 は他の PE から受信する必要がなくなるので，ローカルの情報を使って，担当するブロック $G_i, (s[2] \le i \le e[2])$ の最終計算を行い，計算結果を PE3，PE4 に送信する．その後，PE3，PE4 は，PE2 から受信した $G_i, (s[2] \le i \le e[2])$ を使って，担当するブロックを更新する．ここで，PE3 は他の PE から受信する必要がなくなるので，担当するブロック $G_i, (s[3] \le i \le e[3])$ の最終計算をし，計算結果を PE4 に送信する．最後に，PE4 は PE3 から受信した $G_i, (s[3] \le i \le e[3])$ を使って，担当するブロックを更新する．その後，担当するブロックの最終計算を行う．つまり，l 番目の PE では，以下のような操作が行われる．

- PE1 から $G_i, (s[1] \le i \le e[1])$ を受信
- 受信した $G_i, (s[1] \le i \le e[1])$ を使って，担当するブロックを更新
- $\ldots$
- PE$(l-1)$ から $G_i, (s[l-1] \le i \le e[l-1])$ を受信
- 受信した $G_i, (s[l-1] \le i \le e[l-1])$ を使って，担当するブロックを更新
- 担当するブロックの最終計算を行う

このように，計算と通信を交互に行うことによって，AISM 法の部分並列化を実現する．p 個の PE を使う場合の通信と計算の様子を図 5.5 で示す．ただし，図 5.5 のような処理が部分並列化であるため，すべての PE はすべてのステップで並列に動作できるとは限らない．

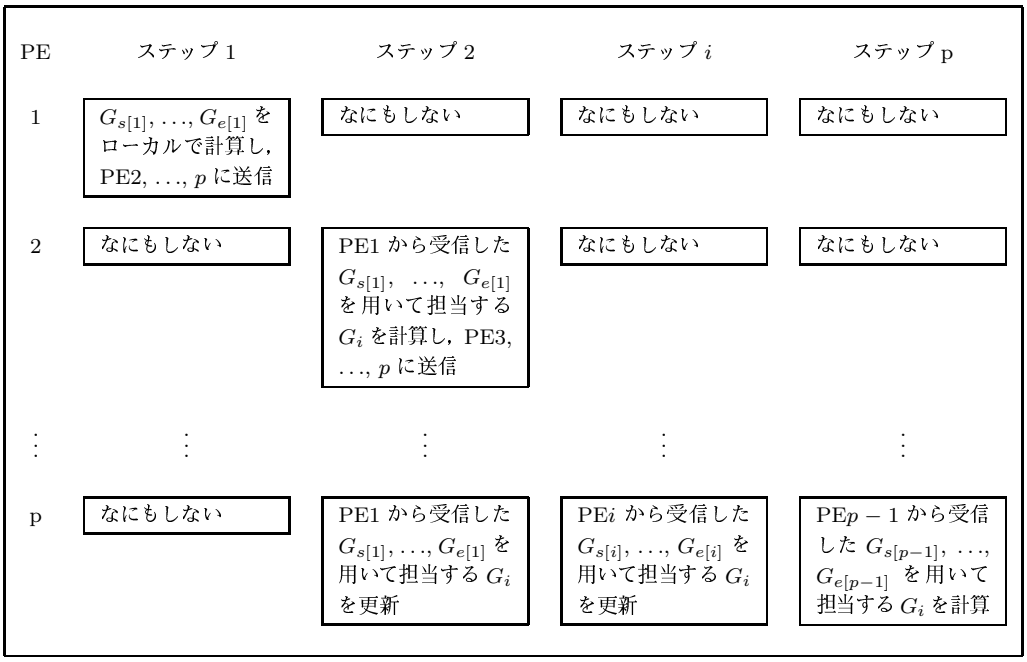


図 5.5 部分並列計算

例えば，ステップ 1 では，PE1 しか動作しない．それから，ステップ 2 以降，PE1 は動作しなくなる．しかし，幾つかの PE が部分的には並列に動作することが可能である．

ここで，図 5.5 のような部分並列化には，1 つの問題がある．それは，PEi が担当するブロックをすべて計算しないと，他の PE に送信できないため，他の PE の待ち時間が長くなる．PE の待ち時間を短縮するために，各 PE が自分の担当するブロックをいくつか，例えば $\tilde{m}$ 個を計算した後，すぐに別の PE に送信することを考える．こうすることによって，他の PE の待ち時間を短縮できる．つまり，すべての計算が終わらないうちに途中で別の PE に送信する．この場合の通信する様子を図 5.6 で示す．ただし，各 PE の担当するブロックの個数が $\tilde{m}$ で割り切れない場合，最後の通信は担当するブロックの個数 $\bmod \tilde{m}$ 個のブロックを送信する．例えば担当するブロックの個数が 501 で $\tilde{m} = 100$ なら，通信は全部で 6 回必要である．最初の 5 回の通信で G_k を 100 個ずつ他 PE に送信する．最後の 6 回目の通信で，余りの 1 個だけを送信することになる．

図 5.5 で示した並列処理と同様に，図 5.6 の処理も，すべての PE がすべてのステップで並列に動作できるとは限らない．しかし，各 PE は，担当するすべてのブロックではなく，$\tilde{m}$ 個のブロックの計算が終わった後，すぐ別の PE に送信することによって，他の PE の待ち時間を短縮できる．本論文では，図 5.6 で示した並列処理を実装する．l 番目の PE の処理は図 5.7 のようになる．ただし，PE の数を p とし，l 番目の PE は $s[l]$ から $e[l]$ までの

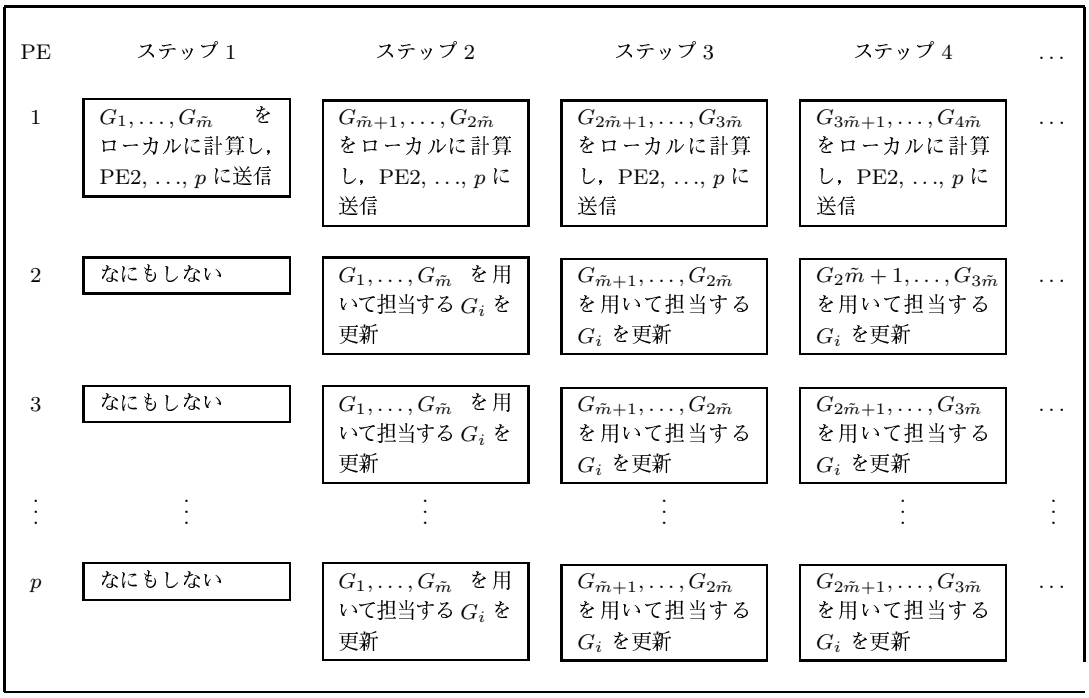


図 5.6 計算の途中で通信する部分並列計算

列ベクトルを担当する．$s[l]$ と $e[l]$ は図 5.4 のように設定されている．この算法はスレッド並列でも実装が可能であるので，この図のタイトルには PE とプロセス両方の表現を用いている．

図 5.7 の 1 行目から 3 行目までは，l 番目の PE の担当している列ベクトル $\boldsymbol{u}_k$, $\boldsymbol{v}_k$ の初期値を設定している．4 行目から 11 行目までは，各 PE が別の PE から計算に必要となる $\boldsymbol{u}_j$, $\boldsymbol{v}_j$, r_j を受信してから，各 PE の担当する $\boldsymbol{u}_k$, $\boldsymbol{v}_k$ を更新している．これらの部分の計算では，1 番目の PE を除いた各 PE は受信が完了しないと 7 行目の受信の過程で待ち状態になる．12 行目から 20 行目までは各 PE の担当する $\boldsymbol{u}_k$, $\boldsymbol{v}_k$ と r_k の最終計算を行っている．この部分の計算では PE 間で通信する必要がない，ローカルに計算することになる．この部分の計算は，式 (5.1) の右辺の最後の項目に相当する．21 行目から 23 行目までは，$\boldsymbol{u}_k$, $\boldsymbol{v}_k$, r_k がそれぞれ $\tilde{m}$ 個だけの計算が終わると，すぐこれらのデータを他の PE に送信している．ただし，他の PE の受信を待たずに，12 行に戻って，次の $\tilde{m}$ 個の $\boldsymbol{u}_k$, $\boldsymbol{v}_k$, r_k を計算する．部分並列化した AISM 法の詳細については，文献 [34] を参照してほしい．

```
 1:  for k = s[l] to e[l] do
 2:    x_k = e_k,   y_k = (a^k − s e_k)^T,   u_k = x_k,   v_k = y_k
 3:  endfor
 4:  for i = 1 to l − 1 do
 5:    for k = s[i] to e[i] do
 6:      if mod(k, m̃) = 1 then
 7:        Receive u_j, v_j, r_j, (j = k, ..., k + m̃ − 1) from PE_i
 8:      endif
 9:      u_k = u_k − {(v_i)_k / (sr_i)}u_i,    v_k = v_k − {(q_k, u_i) / (sr_i)}v_i
10:    endfor
11:  endfor
12:  for k = s[l] to e[l] do
13:    for i = s[l] to k − 1 do
14:      u_k = u_k − {(v_i)_k / (sr_i)}u_i,    v_k = v_k − {(q_k, u_i) / (sr_i)}v_i
15:    endfor
16:    for i = 1 to i = n do
17:      if |(u_k)_i| < tolU   dropoff |(u_k)_i|
18:      if |(v_k)_i| < tolV   dropoff |(v_k)_i|
19:    endfor
20:    r_k = 1 + (v_k)_k/s
21:    if mod(k, m̃) = 0 then
22:      Send u_i, v_i, r_i(i = k − m̃ + 1, ..., k) to PE_{l+1}, ..., PE_p
23:    endif
24:  endfor
```

図 5.7 AISM 法の部分並列化の算法 (l 番目の PE もしくはプロセスの場合)

5.3 数値実験

本数値実験は，3 台のノードで構築した PC クラスタを使用する．これらのノードはネットワークで接続した．数値実験の環境は以下のようになる．

ノード: IBM Xseries 346 (× 3)

PE: Pentium Xeon 3.6GHz (× 2)

メモリ容量: 1 ノードにつき 1GB

OS: Linux Fedora Core 4

コンパイラ: gcc 4.0.0 (オプション一切なし)

通信ライブラリ: LAM/MPI 7.1.1 [35]

本節で行われた数値実験の内容は以下のようになっている．

1. AISM 法による前処理の計算で，通信回数を変化させて，最適なブロック数 $\tilde{m}$ を決定する．ILU 分解における最適な $\tilde{m}$ の値も同様に決定する．
2. AISM 法に現れる行列 U, V の非ゼロ要素数を計測する．
3. PE 台数を 1，2，4，6 台と変化させて，前節で述べた AISM 法の部分並列化を実装したときの並列度を計測する．
4. AISM 法で計算した近似逆行列を前処理行列として用いたときの GMRES(m) 法 [42] の残差ノルムの収束の評価を行う．

実験項目 (3) を除いて，すべての実験は 6 台の PE で行われた．また，実験項目 (4) の残差ノルムの収束の評価は，AISM 法による前処理行列と MR 法もしくは ILU 分解による前処理行列を GMRES(m) 法 [42] に適用して，それぞれの前処理行列としての性能を比較した．数値例として取り扱う問題の係数行列は，実数かつ非対称行列である．

連立 1 次方程式の初期近似解をゼロベクトルとし，GMRES(m) 法の収束判定条件を次式のように設定する．

$$\|\boldsymbol{r}_i\|_2 \,/\, \|\boldsymbol{b}\|_2 < \ 1.0 \times 10^{-12} \tag{5.2}$$

ただし，$\boldsymbol{r}_i$ は GMRES(m) 法の i 回目の反復における残差ベクトルである．

表 5.1 数値例 1: AISM 法の一回送信するブロック数 $\tilde{m}$ と計算時間の関係

(a). tolU, tolV = 0.1

通信回数	1	2	4	8	16	32	64	128
$\tilde{m}$	6144	3072	1536	768	384	192	96	48
計算時間 (秒)	84.0	63.0	59.0	58.0	57.0	57.0	57.0	56.0

(b). tolU, tolV = 0.01

通信回数	1	2	4	8	16	32	64	128
$\tilde{m}$	6144	3072	1536	768	384	192	96	48
計算時間 (秒)	591.0	460.0	405.0	392.0	390.0	389.0	389.0	384.0

表 5.2 数値例 1: ILU(2) 法の一回送信するブロック数 $\tilde{m}$ と計算時間の関係

通信回数	1	2	4	8	16	32	64	128
$\tilde{m}$	6144	3072	1536	768	384	192	96	48
計算時間 (秒)	0.42	0.41	0.41	0.42	0.42	0.44	0.46	0.52

AISM 法に現れる行列 U，V の非ゼロ要素の切り捨てを行う閾値 tolU, tolV は 0.1 もしくは 0.01 の値を使用した．また，スカラ s について，Bru ら [8] に基づいた $s = 1.5 \|A\|_\infty$ の他に，$1.5 \|A\|_\infty \times 10^1$ ，$1.5 \|A\|_\infty \times 10^2$ と $1.5 \|A\|_\infty \times 10^3$ を利用して，s の変化による速度向上や残差ノルムの収束への影響を計測した．ここで，便宜上 $\tilde{s} = s \ / \ (1.5 \|A\|_\infty)$ を定義し，$\tilde{s}$ を用いて s を変化させる．

MR 法には，非ゼロ要素の切り捨てを行う閾値を tol とし，0.1 もしくは 0.01 の値を利用した．MR 法の反復の初期値となる行列には単位行列 I を使用し，内部反復回数 imax は 1 から 3 の値を利用した．ILU 分解では，フィルインを全く許さない ILU(0) と，非ゼロ要素の両サイドにそれぞれ 1，または 2 個の要素のフィルインを許す ILU(1)，ILU(2) の 3 通りの方法により係数行列の不完全 LU 分解を行った．

また，データの割当てについて，MR 法による前処理行列の計算では，前処理行列の列ベクトルを各 PE に均等に割当てた．ILU 分解による前処理行列では，行列 L と U の行ベクトルをブロック分割して各 PE に均等に分散することにした．AISM 法は前節の第 5.2.1 小節で記述した手法で割当てることにした．

[数値例 1] 正方領域 $\Theta = [0,\ 1]^2$ における以下のような偏微分方程式の境界値問題を考える [44] .

$$
\begin{aligned}
-\frac{\partial}{\partial x}\left[\{\exp(-xy)\}u_x\right] - \frac{\partial}{\partial y}\left[\{\exp(xy)\}u_y\right] + 10.0(u_x + u_y) - 60.0u &= f(x,y) \\
u(x,\ y)|_{\partial\Theta} &= 1 + xy
\end{aligned}
$$

表 5.3 数値例 1: AISM 法による前処理行列の非ゼロ要素数

tolU tolV	$\tilde{s}$							
	10^0		10^1		10^2		10^3	
	U	V	U	V	U	V	U	V
0.1	138004	155308	138004	943205	138004	3171837	138004	7027265
0.01	1051329	13164320	1051329	35531595	1051329	66748783	1051329	103157063

ただし，$f(x,y)$ は厳密解が $u(x,y) = 1 + xy$ となるように決定する．この偏微分方程式を 5 点中心差分で離散化した．格子点数は 192×192 である．従って，得られたの連立 1 次方程式の次元は 36,864 である．連立 1 次方程式の係数行列 A は 5 重対角行列であり，各対角成分の値はすべて異なっている．

最初に，ブロック数 $\tilde{m}$ を変化させて AISM 法により近似逆行列の計算時間を計測した．その結果を表 5.1 に示す．$\tilde{m}$ が小さいとき，PE の待ち時間を短縮できるが，通信回数も増大することになる．従って，$\tilde{m}$ を小さくしても，AISM 法の計算時間が必ず改善できるとは限らない．表 5.1 より閾値が tolU, tolV $= 0.1$ と tolU, tolV $= 0.01$ のいずれの場合も，通信回数が 128 回のとき，つまり $\tilde{m} = 48$ のとき，AISM 法による前処理行列の計算時間は一番少なかった．同様にして，ILU 分解で式 (4.1) と式 (4.2) の計算にかかる時間を計測した．その結果を表 5.2 に示す．表 5.2 から，通信回数が 4 回，つまり $\tilde{m} = 1536$ のとき計算時間は最小になったことがわかる．この事実を元に，残差ノルムの収束評価を行う数値実験では，AISM 法と ILU(2) 法の一回で送信するブロック数は，それぞれ $\tilde{m} = 48,\ 1536$ の値を利用した．

次に，s の値を変化させ，AISM 法から得られた行列 $U,\ V$ の非ゼロ要素数を計測し，その結果を表 5.3 に示す．表 5.3 から，V の非ゼロ要素は s に比例して増加する傾向が見られた．特に，s が 3 桁違うと非ゼロ要素数は最大で約 10 倍の差がみられた．従って，$U,\ V$ に必要なメモリの容量は s の大きさに比例することになる．また，s の変化による並列効果への影響を調べるために，$\tilde{s} = 10^0,\ 10^1$ のケースを取り上げて，PE 台数を 1 から 6 まで変化させて，台数効果を計測した．ただし，通信回数が 128 回となるようにブロック数の値 $\tilde{m}$ を設定した．

表 5.4 の計測結果から，PE を 6 台使用したときの台数効果は 4 台までのときに比べて，低下しているものの，約 5.6 倍になって，理想の 90% 程度の速度向上を得ることができた．また，$\tilde{s} = 10.0$ の場合は $\tilde{s} = 1.0$ のときに比べて，PE 数が 4 台と 6 台のときに速度は若干向上している．これは，V の非ゼロ要素数が増加したため，通信量と比べて計算量が相対的に増えたためである．

表 5.4 数値例 1: s の変化による台数効果への影響 (tolU, tolV $= 0.01$)

PE 数	$\tilde{s}$	
	10^0	10^1
1	1.0	1.0
2	1.9	1.9
4	3.8	3.9
6	5.6	5.8

表 5.5 数値例 1: 計算時間と反復回数 (time: 計算時間 (秒), iter: 反復回数)

前処理の種類	前処理行列の計算	非定常反復法					
		GMRES(20)		GMRES(30)		GMRES(40)	
	time	time	iter	time	iter	time	iter
なし	0.0	–	–	–	–	–	–
AISM 法 (tolU = tolV $= 0.1$, $\tilde{s} = 10^0$)	56.0	–	–	514.0	12125	289.0	5839
AISM 法 (tolU, tolV $= 0.1$, $\tilde{s} = 10^1$)	142.0	158.0	338	164.0	453	165.0	473
AISM 法 (tolU, tolV $= 0.01$, $\tilde{s} = 10^0$)	384.0	452.0	1160	420.0	565	409.0	382
AISM 法 (tolU, tolV $= 0.01$, $\tilde{s} = 10^1$)	1461.0	1470.0	57	1468.0	39	1467.0	33
ILU(0) 法	0.21	–	–	–	–	171.0	3066
ILU(1) 法	0.31	–	–	63.0	1044	78.0	1313
ILU(2) 法	0.41	–	–	38.0	625	41.0	588
MR 法 (tol $= 0.1$, imax $= 3$)	173.0	–	–	–	–	–	–
MR 法 (tol $= 0.1$, imax $= 2$)	100.0	–	–	–	–	–	–
MR 法 (tol $= 0.1$, imax $= 1$)	45.0	–	–	–	–	–	–
MR 法 (tol $= 0.01$, imax $= 3$)	189.0	–	–	–	–	–	–
MR 法 (tol $= 0.01$, imax $= 2$)	103.0	–	–	–	–	–	–
MR 法 (tol $= 0.01$, imax $= 1$)	45.0	–	–	–	–	214.0	6021

–: 2 時間以内に残差ノルムが収束しなかった場合
imax: MR 法の反復回数

最後に，6 台の PE を用いて，AISM 法，ILU 分解と MR 法で求めた前処理行列を GMRES(m) 法に適用し，それぞれの前処理行列としての性能を比較した．表 5.5 には GMRES(m) 法の残差ノルムが式 (5.2) の収束判定条件を満たすまでの計算時間と反復回数を示している．ただし，2 列目には前処理行列の計算時間を意味する．GMRES(m) 法の計算時間には，前処理行列の計算時間が含まれている．

MR 法による前処理行列を適用した場合，残差ノルムを 2 時間以内に収束させることができたのは，tol $= 0.1$ かつ imax $= 1$ のときの GMRES(40) 法だけであった．この前処理行列の計算では，imax を増加させても前処理行列の近似度が上がらず，残差ノルムの収束を改善させることができなかった．ILU 分解による前処理を適用すると AISM 法を適用したときより，早い時間で前処理行列を求めることができる．残差ノルムの収束が良くなるケースもある．従って，この数値例から，ILU 分解が AISM 法より前処理行列として優れ

表 5.6 数値例 2: AISM 法のブロック数と計算時間の関係

(a-1). ecl32, tolU, tolV = 0.1

通信回数	1	2	4	8	16	32	64	128
$\tilde{m}$	8666	4333	2167	1084	542	271	136	68
計算時間 (秒)	310.0	230.0	216.0	210.0	207.0	206.0	209.0	209.0

(a-2). ecl32, tolU, tolV = 0.01

通信回数	1	2	4	8	16	32	64	128
$\tilde{m}$	8666	4333	2167	1084	542	271	136	68
計算時間 (秒)	1606.0	1287.0	1144.0	1075.0	1047.0	1038.0	1042.0	1049.0

(b-1). af23560, tolU, tolV = 0.1

通信回数	1	2	4	8	16	32	64	128
$\tilde{m}$	3927	1964	982	491	246	123	62	31
計算時間 (秒)	1835.0	1465.0	1284.0	1185.0	1175.0	1223.0	1228.0	1234.0

た性能を発揮することがある．しかし，GMRES(m) 法のリスタート周期が小さいと収束できず，安定性に欠けることがあった．それに対して，AISM 法による行列の前処理を適用した場合，GMRES(20) 法の 1 ケースを除いて，すべての場合で残差ノルムは収束した．AISM 法による前処理行列の計算に時間がかかるものの，多くの場合で，残差ノルムは安定して収束している．従って，AISM 法で計算した前処理行列の場合，残差ノルムの収束の安定性が一番良いといえる．

さらに，表 5.4 のように PE 台数を変化させて，AISM 法による近似逆行列前処理を用いた GMRES(40) 法の計算時間を計測して，その台数効果を測定した．ただし，tolU, tolV = 0.01 かつ $\tilde{s} = 10^0$ とする．その結果，台数効果は 表 5.4 の $\tilde{s} = 10^0$ の場合とまったく同じ結果となった．これは，GMRES(40) 法の計算時間 409 秒のうち，前処理行列の計算時間が 384 秒であり，総計算時間の 90%以上をしめるため，同じ結果になったと考えることができる．

[数値例 2] Florida Sparse Matrix Collection [47] のデータベースにある行列 “ecl32”, “af23560” を係数行列とする連立 1 次方程式を考える．厳密解の要素がすべて 1.0 となるように連立 1 次方程式の右辺ベクトル $\boldsymbol{b}$ を設定した．行列 “ecl32”, “af23560” の次元数と非ゼロ要素数はそれぞれ “ecl32” が 51993, 347097 であり，“af23560” が 23560, 460456 である．非ゼロ要素は数値例 1 で扱ったものより不規則に配置されている．

数値例 2 の係数行列では，次元数 n が必ずしも PE 台数 $p = 6$ で割り切れるとは限らない．この場合，各 PE の担当する列ベクトルは図 5.4 のように決定する．一回に送信するブロック数 $\tilde{m}$ について，第 5.2.2 小節で述べたように，各 PE の担当するブロックの個数が $\tilde{m}$ で割り切れない場合，最後の通信は担当するブロックの個数 $\bmod \tilde{m}$ 個のブロックを送信することになる．以上のことを踏まえて AISM 法と ILU(2) 法の最適なブロック数 $\tilde{m}$ の計

表 5.7 数値例 2: ILU(2) 法のブロック数と計算時間の関係

(a). ecl32

通信回数	1	2	4	8	16	32	64	128
$\tilde{m}$	8666	4333	2167	1084	542	271	136	68
計算時間 (秒)	2.17	2.14	2.14	2.13	2.14	2.17	2.23	2.34

(b). af23560

通信回数	1	2	4	8	16	32	64	128
$\tilde{m}$	3927	1964	982	491	246	123	62	31
計算時間 (秒)	2.12	2.12	2.11	2.10	2.10	2.10	2.12	2.20

表 5.8 数値例 2: AISM 法による前処理行列の非ゼロ要素数

(a). ecl32

tolU tolV	$\tilde{s}$							
	10^0		10^1		10^2		10^3	
	U	V	U	V	U	V	U	V
0.1	204069	2255976	204069	3762618	204069	5604401	204069	7775226
0.01	1535930	11331526	1535930	17858501	1535930	25564994	1535930	33866053

(b). af23560

tolU tolV	$\tilde{s}$							
	10^0		10^1		10^2		10^3	
	U	V	U	V	U	V	U	V
0.1	551740	32207303	551740	61057101	551740	93002049	551740	124318919

測結果をそれぞれ 表 5.6 と 表 5.7 に示す.

“ecl32” の場合には，AISM 法では tolU, tolV $= 0.1$ と tolU, tolV $= 0.01$ のいずれの場合も通信回数が 32 回，つまり $\tilde{m} = 271$ のとき計算時間が最小となった．ILU(2) 法では $\tilde{m} = 1084$ のとき最小となった．“af23560” の場合には，AISM 法は tolU, tolV $= 0.01$ のとき，2 時間以内に前処理行列の計算が終わらなかったが，tolU, tolV $= 0.1$ かつ $\tilde{m} = 246$ のときに計算時間は最小となった．ILU(2) 法の計算時間が最小となったのは，$\tilde{m} = 123$ のときであった．従って，残差ノルムの収束評価の実験で，AISM 法による前処理行列を用いるときは，“ecl32” の場合，AISM 法に $\tilde{m} = 271$ を利用し，ILU 分解に $\tilde{m} = 1084$ を使用した．“af23560” の場合，AISM 法と ILU 分解に，それぞれ $\tilde{m} = 246$，$\tilde{m} = 123$ に設定した．

次に，これら 2 つの行列について，数値例 1 同様に s の値を変化させて， U, V の非ゼロ要素数を計測した．ただし，“af23560” の tolU, tolV $= 0.01$ では，2 時間以内で計算が完了しなかったので計測結果の記述を省略する.

表 5.9 数値例 2: “af23560” の s の変化による台数効果への影響 (tolU, tolV $= 0.1$)

PE 数	$\tilde{s}$	
	10^0	10^1
1	1.0	1.0
2	1.7	1.9
4	2.8	3.0
6	3.8	4.0

表 5.8 から，これらの例においても s に比例して V の非ゼロ要素数が増加する傾向があることがわかる．“af23560” では $\tilde{s} = 10^3$ のとき $\tilde{s} = 10^0$ のときの約 4 倍の非ゼロ要素数が得られた．ここで，非ゼロ要素を 8 バイトの倍精度の浮動小数点型で記憶したとすると，V の非ゼロ要素を記憶するために，必要となる記憶容量は $8 \times 124318919 = 994551352$，約 994MB となる．本数値実験では，3 つのノードを使用しているので，1 ノードあたり V だけを記憶するために，約 331MB 程度の記憶領域が必要となる．このことを考慮すると，必ずしも s を大きくとることが有効とは限らない．ここで，$\tilde{s}$ が 10^0, 10^1 のときの台数効果を表 5.9 に示す．

この例で得られた台数効果は，数値例 1 より少ないものの，6 台使用したとき 4 倍近い速度向上が観測された．つまり，理想の 2/3 程度の性能が得られた．また，$\tilde{s} = 10.0$ の場合は $\tilde{s} = 1.0$ のときに比べて，台数効果は少し改善されている．これは，数値例 1 と同様に，V の非ゼロ要素数が増加したため，通信量と比べて計算量が相対的に増えたためである．

最後に，6 台の PE を用いて，AISM 法，ILU 分解と MR 法の前処理行列としての性能を比較した．表 5.10 には “ecl32” と “af23560” を係数行列とする連立 1 次方程式の近似解が収束判定条件を満たすまでの GMRES(m) 法の計算時間と反復回数を示す．

“ecl32” の場合には，MR 法による前処理は収束しなかった．ILU 分解による前処理では，“ecl32” に ILU(0) 法を適用した GMRES(40) 法の場合のみ残差ノルムは収束した．AISM 法で計算した前処理行列を使用すると，前処理行列の計算に時間が掛かったものの，閾値 tolU, tolV の値を 0.01 に設定したとき，すべての GMRES(m) 法は収束した．GMRES(m) 法の残差ノルムの収束の安定性を考慮すると，AISM 法による前処理行列の計算コストが割高となっても AISM 法を用いる価値があるといえる．

次に，“af23560” の場合には，閾値が 0.01 では計算が 2 時間以内に終わらず，前処理を GMRES(m) 法に適用しなかったが，閾値が 0.1 のときにはすべての GMRES(m) 法で残差ノルムの収束を確認することができた．それに対して，ILU 分解と MR 法を適用する場合は収束することはなかった．従って，このように非ゼロ要素の分布が比較的不規則な問題に対して，ILU 分解による前処理行列よりも AISM 法による近似逆行列を使う方が有効で

表 5.10 数値例 2: 計算時間と反復回数 (time: 計算時間 (秒), iter: 反復回数)

(a). ecl32

前処理の種類	前処理行列の計算	非定常反復法					
		GMRES(20)		GMRES(30)		GMRES(40)	
	time	time	iter	time	iter	time	iter
なし	0.0	–	–	–	–	–	–
AISM 法 (tolU, tolV $= 0.1$, $\tilde{s} = 10^0$)	208.0	–	–	–	–	–	–
AISM 法 (tolU, tolV $= 0.1$, $\tilde{s} = 10^1$)	290.0	–	–	–	–	–	–
AISM 法 (tolU, tolV $= 0.01$, $\tilde{s} = 10^0$)	1038.0	1128.0	790	1090.0	408	1065.0	231
AISM 法 (tolU, tolV $= 0.01$, $\tilde{s} = 10^1$)	1455.0	1570.0	819	1519.0	415	1485.0	221
ILU(0) 法	0.48	–	–	–	–	1188.0	22553
ILU(1) 法	1.14	–	–	–	–	–	–
ILU(2) 法	2.13	–	–	–	–	–	–
MR 法 (tol $= 0.1$, imax $= 3$)	402.0	–	–	–	–	–	–
MR 法 (tol $= 0.1$, imax $= 2$)	238.0	–	–	–	–	–	–
MR 法 (tol $= 0.1$, imax $= 1$)	99.0	–	–	–	–	–	–
MR 法 (tol $= 0.01$, imax $= 3$)	544.0	–	–	–	–	–	–
MR 法 (tol $= 0.01$, imax $= 2$)	277.0	–	–	–	–	–	–
MR 法 (tol $= 0.01$, imax $= 1$)	99.0	–	–	–	–	–	–

(b). af23560

前処理の種類	前処理行列の計算	非定常反復法					
		GMRES(20)		GMRES(30)		GMRES(40)	
	time	time	iter	time	iter	time	iter
なし	0.0	–	–	–	–	–	–
AISM 法 (tolU, tolV $= 0.1$, $\tilde{s} = 10^0$)	1175.0	1344.0	759	1328.0	621	1302.0	549
AISM 法 (tolU, tolV $= 0.1$, $\tilde{s} = 10^1$)	2304.0	2562.0	740	2526	649	2478.0	544
ILU(0) 法	0.62	–	–	–	–	–	–
ILU(1) 法	1.31	–	–	–	–	–	–
ILU(2) 法	2.10	–	–	–	–	–	–
MR 法 (tol $= 0.1$, imax $= 3$)	180.0	–	–	–	–	–	–
MR 法 (tol $= 0.1$, imax $= 2$)	111.0	–	–	–	–	–	–
MR 法 (tol $= 0.1$, imax $= 1$)	48.0	–	–	–	–	–	–
MR 法 (tol $= 0.01$, imax $= 3$)	254.0	–	–	–	–	–	–
MR 法 (tol $= 0.01$, imax $= 2$)	126.0	–	–	–	–	–	–
MR 法 (tol $= 0.01$, imax $= 1$)	48.0	–	–	–	–	–	–

–: 2 時間以内に計算が完了しなかった場合

imax: MR 法の反復回数

表 5.11 数値例 3: AISM 法のブロック数と計算時間の関係

(a). tolU, tolV = 0.1

通信回数	1	2	4	8	16	32	64	128
$\tilde{m}$	40000	20000	10000	5000	2500	1250	625	313
計算時間 (秒)	1512.0	1146.0	1055.0	1034.0	1030.0	1027.0	1027.0	1028.0

(b). tolU, tolV = 0.01

通信回数	1	2	4	8	16	32	64	128
$\tilde{m}$	40000	20000	10000	5000	2500	1250	625	313
計算時間 (秒)	1513.0	1146.0	1056.0	1034.0	1031.0	1027.0	1026.0	1027.0

表 5.12 数値例 3: ILU(2) 法のブロック数と計算時間の関係

通信回数	1	2	4	8	16	32	64	128
$\tilde{m}$	40000	20000	10000	5000	2500	1250	625	313
計算時間 (秒)	0.78	0.76	0.77	0.79	0.88	1.07	1.49	2.85

あると考えられる.

[数値例 3] AISM 法の性能を調べるために，ILU 分解に適する次のようなブロック対角行列

$$A = \mathrm{diag}\,[A_1,\, A_2,\, \ldots,\, A_{\tilde{n}}] \in \boldsymbol{R}^{n\times n} \tag{5.3}$$

を係数とする連立 1 次方程式を考える [20]. ただし，$n = 240,000$，$\tilde{n} = 120,000$ ，

$$A_j = \begin{bmatrix} \lambda_{re,j} & \lambda_{im,j} \\ -\lambda_{im,j} & \lambda_{re,j} \end{bmatrix} \in \boldsymbol{R}^{2\times 2}, \quad j = 0,\ 1, \ldots, \tilde{n}$$

である. 厳密解の要素は，$[0,\ 1]$ の範囲の乱数で決定し右辺 $\boldsymbol{b}$ を設定した. $\lambda_{im,j}$ は $[-1,\ 1]$ の範囲の乱数で設定し，$\lambda_{re,j}$ は $[10^{-3},\ 10^{-2}]$ の範囲の乱数を利用した.

この数値例の係数行列は三重対角行列であり，フィルイン現象を起こさずに LU 分解することが可能である [49]. 従って，このような係数行列に ILU 分解を行うと，完全 LU 分解になり，前処理行列を適用する時点で解が求まる. つまり，GMRES(m) 法は 1 回の反復で解を求めることができる. 本数値例は，この問題を取り上げて，ILU 分解に適する問題に対して，AISM 法の性能を調べる.

まず，他の数値例同様，最初に通信回数を変化させて，6 台の PE で AISM 法と ILU 分解による前処理行列の計算時間を計測した. その結果を表 5.11 と 表 5.12 に示す.

AISM 法では $\tilde{m} = 625$ のとき，ILU 分解では $\tilde{m} = 20000$ のときに計算時間が最小になったので，残差ノルムの収束評価の実験においてもブロック数としてこれらの値を用いた.

表 5.13 数値例 3: AISM 法による前処理行列の非ゼロ要素数

tolU tolV	$\tilde{s}$							
	10^0		10^1		10^2		10^3	
	U	V	U	V	U	V	U	V
0.1	359937	479930	359937	479931	359937	479931	359937	479931
0.01	359990	479984	359990	479984	359990	479984	359990	479984

表 5.14 数値例 3: AISM 法による前処理行列の計算の速度向上 (tolU, tolV = 0.01)

PE 数	$\tilde{s}$	
	10^0	10^1
1	1.00	1.00
2	1.39	1.39
4	2.38	2.39
6	3.12	3.14

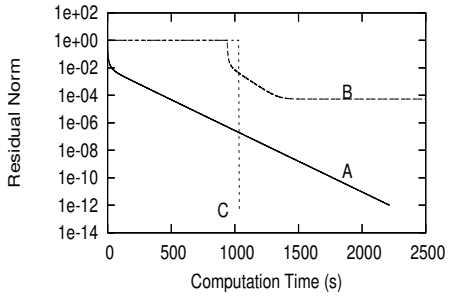
次に，前の 2 つの数値例と同様に，s を変化させ，非ゼロ要素数を計測した．その結果を表 5.13 に示す．この数値例では，s を変化させても V の非ゼロ要素数に大幅な変化は見られなかった．

また，数値例 1 と 2 と同様の手順で PE 台数を変化させて，AISM 法による前処理行列の計算時間を計測した．その結果を表 5.14 に示す．表 5.14 から，この例では 6 台の PE で 3 倍以上の速度向上が得られたことがわかる．

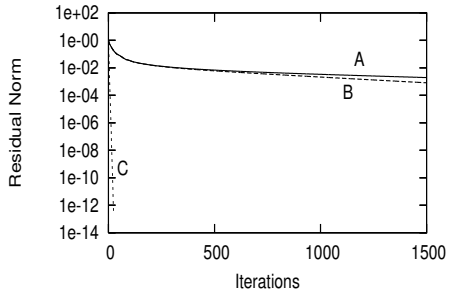
表 5.15 数値例 3: 計算時間と反復回数 (time: 計算時間 (秒), iter: 反復回数)

前処理の種類	前処理行列の計算	非定常反復法					
		GMRES(20)		GMRES(30)		GMRES(40)	
	time	time	iter	time	iter	time	iter
なし	0.0	1891.0	24083	2065.0	23455	2216.0	23131
AISM 法 (tolU, tolV = 0.1, $\tilde{s} = 10^0$)	1027.0	1035.0	24	1034.0	24	1034.0	24
AISM 法 (tolU, tolV = 0.1, $\tilde{s} = 10^1$)	1026.0	1027.0	9	1027.0	9	1027.0	9
AISM 法 (tolU, tolV = 0.01, $\tilde{s} = 10^0$)	1026.0	1034.0	24	1034.0	24	1035.0	24
AISM 法 (tolU, tolV = 0.01, $\tilde{s} = 10^1$)	1027.0	1028.0	9	1028.0	9	1028.0	9
ILU(0) 法	0.76	1.0	1	1.0	1	1.0	1
ILU(1) 法	0.76	1.0	1	1.0	1	1.0	1
ILU(2) 法	0.76	1.0	1	1.0	1	1.0	1
MR 法 (tol = 0.1, imax = 1)	636.0	–	–	–	–	–	–
MR 法 (tol = 0.01, imax = 1)	636.0	–	–	–	–	–	–
MR 法 (tol = 0.1, imax = 2)	1271.0	–	–	–	–	–	–
MR 法 (tol = 0.01, imax = 2)	1274.0	–	–	–	–	–	–

–: 2 時間以内に計算が完了しなかった場合
imax: MR 法の反復回数



(a). 残差ノルム vs. 計算時間



(b). 残差ノルム vs. 反復回数

図 5.8 数値例 3: GMRES(40) 法の残差ノルムの収束する様子，A: 前処理なし, B: MR 法 ($\mathrm{tol} = 0.01,\ \mathrm{imax} = 1$), C: AISM 法 ($\mathrm{tolU},\ \mathrm{tolV} = 0.01,\ \tilde{s} = 1.0$)

最後に，GMRES(m) 法の残差ノルムが収束するまでに要した計算時間と反復回数を 表 5.15 に示す．

この数値例はフィルイン現象を起こさずに LU 分解できるので，GMRES(m) 法では 1 回の反復で解が求まる．このような ILU 分解に非常に適する問題においても，前処理を行わない GMRES(m) 法や MR 法を用いた場合に比べて，AISM 法を用いた場合残差ノルムの収束が改善された．閾値を 0.1 から 0.01 に変化しても，前処理行列の計算時間はほぼ同じであった．これは，異なる閾値を設定したにもかかわらず，表 5.13 で示したように，前処理行列の非ゼロ要素数がほぼ同じになったためであると考えられる．また，s の値を変化させても，非ゼロ要素数が変化しなかったことから，s の値は並列効果に影響を与えなかったことがわかる．残差ノルムの収束に関しては，前処理なしでも収束するが，その反復回数が膨大になっている．また，MR 法による前処理行列を適用しても残差ノルムの収束を改善することはできず，かえって収束を悪化させている．一方，AISM 法で行列の前処理を行う場合，前処理行列の計算だけで 20 分近くかかっているものの，反復回数は前処理なしの場合の約 1000 分の 1 になっているため，総計算時間は前処理を行わない場合より少ない．また，この数値例では s を変化させると反復回数は変わるが，計算時間に大きな変化はみられなかった．

図 5.8 に GMRES(40) 法の残差ノルムの収束する様子を示す．図 5.8 より，前処理を行わなくても残差ノルムは計算時間に対して一定の速度で収束している．AISM 法による前処理を適用すると，残差ノルムは，途中の 1000 秒を過ぎた時点から急激に 1.0×10^{-12} 付近まで下降している．つまり，前処理行列の計算 (約 1000 秒) が終わった後，24 回の反復回数ですぐに収束した．また，MR 法による前処理を適用した場合，前処理行列の計算が

表 5.16 各 PE の稼働率と担当した非ゼロ要素数

(a). 数値例 1, tolU, tolV = 0.01, $\tilde{s} = 10^0$

PE 番号	担当した非ゼロ要素数	計算時間	送受信の時間	同期の時間	稼働率 (%)
0	2125451	36.0	14.0	334.0	9.0
1	2433024	101.0	18.0	265.0	26.0
2	2452044	156.0	33.0	195.0	41.0
3	2363144	212.0	49.0	123.0	56.0
4	2445060	266.0	64.0	54.0	70.0
5	2396926	324.0	60.0	0.0	85.0

(b). 数値例 2, af23560 かつ tolU, tolV = 0.1, $\tilde{s} = 10^0$

PE 番号	担当した非ゼロ要素数	計算時間	送受信の時間	同期の時間	稼働率 (%)
0	2057346	40.0	6.0	1129.0	3.0
1	7026059	242.0	15.0	918.0	21.0
2	11468891	589.0	77.0	509.0	50.0
3	8740500	856.0	210.0	109.0	73.0
4	2627566	771.0	401.0	23.0	56.0
5	838681	752.0	413.0	0.0	66.0

(c). 数値例 3, tolU, tolV = 0.01, $\tilde{s} = 10^0$

PE 番号	担当した非ゼロ要素数	計算時間	送受信の時間	同期の時間	稼働率 (%)
0	139996	11.0	78.0	937.0	1.0
1	139997	180.0	95.0	751.0	17.0
2	139993	342.0	122.0	562.0	33.0
3	139999	484.0	167.0	375.0	47.0
4	139995	628.0	213.0	185.0	62.0
5	139994	768.0	258.0	0.0	75.0

稼働率=計算時間 / (計算時間+送受信の時間+同期の時間)

終了した 900 秒後くらいから残差ノルムが急激に減少し始めたが，1500 秒付近から停滞したため，前処理の効果が現れなかったと言える．

[台数効果の比較] ここで，数値例全体の台数効果を比較するため，各 PE の稼働率と担当した非ゼロ要素数を表 5.16 に示す．ただし，ここの送受信の時間とは，待ち時間を含む各 PE が送，受信に要した時間の合計である．また，1 番目の PE 以外の PE が最初にブロックを受信するまでの受信待ち時間も含まれている．同期の時間とは，各 PE の担当する計算をすべて完了した後，別の PE の計算が終了するまでの待ち時間のことである．稼働率は次の式によって計算される．

稼働率=計算時間 / (計算時間+送受信の時間+同期の時間)

また，担当した非ゼロ要素数とは，各 PE が担当した列ベクトル $\boldsymbol{u}_k$ と $\boldsymbol{v}_k$ に含まれていた非ゼロ要素数の合計値のことである．

表 5.16 から，数値例 1 の各 PE の稼働率は，他の 2 つに比べて高いということが分かった．数値例 2 では，各 PE の計算した非ゼロ要素数の分布は不規則であった．これは，係数行列が数値例 1 と 3 に比べて元々不規則な構造になっているためである．そのため，数値例 2 では各 PE の計算した非ゼロ要素数にはバラつきがあり，PE の稼働率も数値例 1 より低下したと考えられる．また，数値例 3 は数値例 2 ほど非ゼロ要素数のバラつきはなく，数値例 1 同様に各 PE で比較的に均一であった．しかし，数値例 1 と比べて各 PE の計算した非ゼロ要素数は少なかったために，計算が比較的早く終了してしまい，その後の別の PE との同期にかかった時間は数値例 1 よりも多くなったからである．以上のことは，数値例 2 と 3 が数値例 1 ほどの並列効果が出なかった原因と考えられる．

5.4 本章のまとめ

本章では，Naik [36] の手法を用いて，AISM 法による前処理行列としての近似逆行列を求める計算手順を部分並列化する方法を提案した．列ベクトルを各 PE に分割し，これらのベクトルを $\tilde{m}$ 個だけ計算する度に通信を行い，計算と通信の過程を交互に繰り返すことにより部分的な並列化を可能とした．また，このような並列化により一定の速度向上が得られる．本章の数値例では，6 台の PE で，最大 5.8 倍の台数効果が得られた．さらに，提案した手法で計算を行った近似逆行列前処理は，MR 法を並列化して計算したものよりも残差ノルムの収束を改善できることを確認した．

第 6 章

AISM法を用いた2レベル並列計算

第 5 章で提案した Naik [36] の実装を用いた AISM 法の部分並列化は有効な並列手法の 1 つである．しかし，「通信回数が多く，算法の実装はそれほど簡単ではない」という問題点がある．これらの問題を改善するために，2 レベル並列計算に AISM 法を適用することを考える．

2 レベル並列計算とは，グラフ分割を利用して係数行列の要素を並列計算向きに再構成し，並列処理を行う算法である．今，行列 $A \equiv (a_{ij}) \in \boldsymbol{R}^{n\times n}$ と対応する無向グラフを $G =< V, E >$ とする．ただし，$V = \{1, \ldots, n\}$ はグラフに属するノードの集合であり，E はエッジ $\{< i, j > | i, j \in V, a_{ij} \neq 0\}$ の集合である．グラフ分割によってグラフ G をほぼ均等に p 個に分割すると，得られる部分グラフ G_i に属するノードは内部点と境界点に分けられる．内部点とは，ほかの部分グラフ G_j $(j \neq i,\ j = 1, \ldots, p)$ に属する頂点との間に，エッジのないノードである．内部点以外の点は境界点である．ここで，内部点，境界点の順番で再オーダーリングを行い，行列 A にも新しい番号に従って行と列の交換を行うと，行列 A は次式のような羽状の規則正しいブロック行列に変換できる [11].

$$P^T AP = \begin{bmatrix} A_1 & & & B_1 \\ & \ddots & & \vdots \\ & & A_p & B_p \\ C_1 & \cdots & C_p & A_s \end{bmatrix} \tag{6.1}$$

ただし，P は置換行列である．A_i の次元は部分グラフ G_i の内部点の数と一致し，A_s の次元は境界点の総数と一致する．

式 (6.1) の逆行列を次式のように分解することを考える．

$$(P^T AP)^{-1} = \begin{bmatrix} A_1^{-1} & & & E_1 \\ & \ddots & & \vdots \\ & & A_p^{-1} & E_p \\ & & & S^{-1} \end{bmatrix} \begin{bmatrix} I_1 & & & \\ & \ddots & & \\ & & I_p & \\ F_1 & \cdots & F_p & I_s \end{bmatrix} \tag{6.2}$$

ただし，$E_i = -A_i^{-1}B_iS^{-1}$, $F_i = -C_iA_i^{-1}$ であり, $S = A_s - \sum_{i=1}^{p} C_iA_i^{-1}B_i$ はシュールコンプリメントである．I_i と I_s はそれぞれ A_i，A_s と同次元の単位行列である．

式 (6.2) で行列 (P^TAP) 逆行列を計算する際に，右側の $A_i^{-1}(i = 1, \ldots, p)$ は互いに独立に計算できるので，並列計算は可能である．さらに，シュールコンプリメントのローカル部分 $S_i = C_iA_i^{-1}B_i$，$(i = 1, \ldots, p)$ も互いに独立しているので，各々の PE で独立に計算できる．計算する際に，まず，レベル 1 では，各 PE で A_i の逆行列 A_i^{-1} を計算する．次に，各 PE でシュールコンプリメントのローカル部分 $S_i = C_iA_i^{-1}B_i$ を計算する．その次に，各 PE から $C_iA_i^{-1}B_i$ を集めて，シュールコンプリメント S を計算する．最後に，レベル 2 では，シュールコンプリメントの逆行列を計算する．

6.1 AISM 法を用いた 2 レベル並列計算とその実装

6.1.1 AISM 法を用いた 2 レベル並列計算

前節で述べた 2 レベル並列計算に，AISM 法を適用することを考える．式 (6.2) の右側の A_i^{-1} と S^{-1} をそれぞれの逆行列を AISM 法で近似すると，P^TAP の近似逆行列は次式のようになる．

$$(P^TAP)^{-1} \approx M = \begin{bmatrix} M_1 & & & \bar{E}_1 \\ & \ddots & & \vdots \\ & & M_p & \bar{E}_p \\ & & & M_s \end{bmatrix} \begin{bmatrix} I_1 & & & \\ & \ddots & & \\ & & I_p & \\ \bar{F}_1 & \cdots & \bar{F}_p & I_s \end{bmatrix} \tag{6.3}$$

ただし，$M_i \approx A_i^{-1}$，$M_s \approx \bar{S}^{-1}$，$\bar{E}_i = -M_iB_iM_s$，$\bar{F}_i = -C_iM_i$ である．ここで，$\bar{S} = A_s - \sum_{i=1}^{p} C_iM_iB_i$ は近似シュールコンプリメントである．

前小節で述べた事柄と同様に，$M_i(i = 1, \ldots, p)$ は互いに独立に計算できる．さらに，$\bar{S}_i = C_iM_iB_i$，$(i = 1, \ldots, p)$ も各々の PE で独立に計算できる．具体的に，まず，レベル 1 では，各 PE で AISM 法を使って，A_i の近似逆行列 M_i を計算する．次に，各 PE でシュールコンプリメントのローカル部分 $\bar{S}_i = C_iM_iB_i$ を計算する．その次に，各 PE から $C_iM_iB_i$ を集めて，近似シュールコンプリメント $\bar{S}$ を計算する．最後に，レベル 2 では，近似シュールコンプリメントの近似逆行列 M_s を AISM 法で計算する．i 番目の PE での処理は図 6.1 のようになる．この算法はスレッド並列でも実装が可能であるので，図 6.1 のタイトルには PE とプロセス両方の表現を用いている．

データの配置について，A_i，B_i，C_i を i 番目の PE に配置し，PE の数は p と一致する．$\bar{E}_i$，$\bar{F}_i$ は明示的には保存しない．また，本論文では，行列 A_s を全 PE に配置する．各 PE

1: compute $M_i \approx A_i^{-1}$ with AISM method
2: compute $\bar{S}_i = C_i M_i B_i$
3: gather $\bar{S}_j$, $(j = 1, \ldots, p\ j \neq i)$
4: compute $\bar{S} = A_s - \sum_{j=1}^{p} \bar{S}_j$
5: compute $M_s \approx \bar{S}^{-1}$ with AISM method

図 6.1 AISM 法を用いた 2 レベル並列計算 (i 番目の PE もしくはプロセスの場合)

では，ほかの PE から受信した $C_i M_i B_i$ を利用して，近似シュールコンプリメントとその近似逆行列を各 PE で計算し保存する．このことにより重複計算をすることになるが，計算した近似シュールコンプリメント $\bar{S}$ とその近似逆行列 M_s を他の PE に送信する必要はなくなる．

6.1.2 GMRES(m) 法の並列実装

AISM 法を用いた 2 レベル並列計算で計算した近似逆行列を GMRES(m) 法の前処理行列とする場合の GMRES(m) 法の並列実装について考える．

AISM 法を用いた 2 レベル並列計算で前処理行列を計算した後，i 番目の PE は A_i, B_i, C_i, A_s, M_i, $\bar{S}$, M_s を保有する．これに合わせて，GMRES(m) に現れるベクトル $x \in \boldsymbol{R}^n$ も $(x_1, x_2, \ldots, x_p, x_s)^T$ に分割し，x_i と x_s を i 番目の PE に配置する．ただし，$x_1, x_2, \ldots, x_p, x_s$ の次元はそれぞれ $A_1, A_2, \ldots, A_p, A_s$ の次元と一致する．また，$\bar{E}_i = -M_i B_i M_s$，$\bar{F}_i = -C_i M_i$ とベクトルの積を計算するための行列 M_i, B_i, M_s, C_i は各 PE に保存されるので，$\bar{F}_i$ と $\bar{E}_i$ を明示的に保存する必要はない．

行列 $P^T AP$ とベクトルの積は式 (6.4) で計算する．

$$\begin{bmatrix} A_1 & & & B_1 \\ & \ddots & & \vdots \\ & & A_p & B_p \\ C_1 & \cdots & C_p & A_s \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_p \\ x_s \end{bmatrix} = \begin{bmatrix} A_1 x_1 + B_1 x_s \\ \vdots \\ A_p x_p + B_p x_s \\ \sum_{i=1}^{p} C_i x_i + A_s x_s \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_p \\ y_s \end{bmatrix} \tag{6.4}$$

並列計算の際に，まず，i 番目の PE で，y_i と $C_i x_i$ を計算する．その次に，計算した $C_i x_i$ を各 PE に通信し，各 PE では，受信した $C_i x_i$ を使って，y_s を計算する．計算が終わった後，i 番目の PE は，結果ベクトル y の部分ベクトル y_i, y_s を持つことになる．

ベクトル x と y の内積 $\alpha = \sum_{i=1}^{p} < x_i, y_i > + < x_s, y_s >$ を求める場合，まず，i 番目の PE で，$< x_i, y_i >$ を計算する．その次に，ある PE，例えば 1 番目の PE で $< x_s, y_s >$ を計

算する．最後に，各 PE での計算結果を集計した結果が α になる．

前処理行列 M とベクトルの積は式 (6.5) で計算する．

$$\begin{bmatrix} M_1 & & & \bar{E}_1 \\ & \ddots & & \vdots \\ & & M_p & \bar{E}_p \\ & & & M_s \end{bmatrix} \begin{bmatrix} I_1 & & & \\ & \ddots & & \\ & & I_p & \\ \bar{F}_1 & \cdots & \bar{F}_p & I_s \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_p \\ x_s \end{bmatrix} = \begin{bmatrix} M_1 x_1 + \bar{E}_1 z_s \\ \vdots \\ M_p x_1 + \bar{E}_p z_s \\ M_s z_s \end{bmatrix}, \tag{6.5}$$

ただし，$z_s = \sum_{i=1}^{p} \bar{F}_i x_i + x_s$．並列計算の際に，まず $\bar{F}_i x_i$ を PEi で独立に計算する．その後，$\bar{F}_i x_i$ を別の PE に送信する．各 PE では，受信した $\bar{F}_i x_i$ を使って，z_s を計算する．そのあと，各 PE では，$M_i x_i + \bar{E}_i z_s$ と $M_s z_s$ を計算する．

6.2 数値実験

本数値実験は，SGI 社の共有メモリ型並列計算機 Origin 2400 を利用し，以下の条件にて行った．

PE: MIPS R12000(300MHz) × 16

メモリ: 8GB

OS: IRIX 6.5.22

コンパイラ: gcc version 2.8.1

通信ライブラリ: MPI-1.2

プログラム言語: C 言語

計算精度: 倍精度

数値実験の内容は以下のようになっている．

1. pmetis [26] を用いてグラフ分割を行い，係数行列を式 (6.1) のような羽状のブロック行列に変形する．ただし，p の値を 1, 2, 4, . . . , 16 とする．
2. 並列実装を行い，PE 台数を 1, 2, 4, . . . , 16 台とし，速度向上を計測する．
3. 16 台の PE を用いて，AISM 法で計算した近似逆行列を前処理行列として用いたときの GMRES(m) 法 [42] の残差ノルムの収束の評価を行う

表 6.1 グラフ分割の結果

p	A_i の次元	A_s の次元
2	18239/18239	386
4	9033/9024/9014/9013	780
6	5986/5892/5986/5992/5819/5975	1214
8	4384/4457/4365/4465/4439/4380/4448/4360	1566
10	3486/3557/3505/3446/3513/3456/3562/3478/ 3411/3536	1914
12	2864/2960/2898/2941/2852/2910/2846/2908/ 2926/2948/2843/2904	2064
14	2431/2522/2473/2489/2470/2467/2406/2514/ 2412/2466/2444/2516/2415/2477	2362
16	2173/2110/2198/2157/2155/2110/2202/2159/ 2156/2182/2083/2095/2168/2199/2169/2088	2460

ただし，実験項目 (2) では，利用する PE 台数と同じ p の値を持つ実験項目 (1) の結果を使用した．また，実験項目 (3) の残差ノルムの評価は，第 5.3 節の数値実験と同様に，Sherman-Morrison 法による前処理行列と MR 法もしくは ILU 分解による前処理行列を GMRES(m) 法 [42, 44] に適用して，それぞれの前処理行列の性能を比較した．GMRES(m) 法のリスタート周期は 20，30，40 に設定し，反復回数は Krylov 部分空間の次元が 1 つ増加する度に 1 回と数える．計算時間については，MPI_Wtime() 関数で求めた値を秒単位で表示する．

残差ノルムの収束評価に関しては，連立 1 次方程式の初期近似解をゼロベクトルとし，GMRES(m) 法の収束判定条件を

$$\frac{\|\boldsymbol{r}_k\|_2}{\|\boldsymbol{b}\|_2} \leq 1.0 \times 10^{-12} \tag{6.6}$$

と設定する．ただし，最大反復回数 20000 以内で条件 (6.6) を満たせなければ反復を打ち止めにする．

ここで，AISM 法のパラメータ s は，Bru ら [8] に基づき $s = 1.5\times \|A\|_\infty$ の値を利用した．また，行列 U，V の非ゼロ要素の切り捨てを行う閾値は，tolU = tolV = 0.1 とする．近似シュールコンプリメントの計算については，非ゼロ要素の切り捨て処理は行わない．

MR 法の反復の初期値となる行列には単位行列 I を選択し，この前処理計算法の反復回数を imax とし 1 から 3 の値を利用した．MR 法には，非零要素の切り捨てを行う閾値を tol とし，0.1 もしくは 0.01 の値を利用した．

ILU 分解では，フィルインを全く許さない ILU(0) と，非零要素の両サイドにそれぞれ 1 ないしは 2 個の要素のフィルインを許す ILU(1)，ILU(2) の 3 通りの方法により前処理行列を計算した．数値例として取り扱う問題の係数行列は実数かつ非対称とした．

表 6.2 AISM 法の速度向上

p	速度向上
2	4.46
4	18.36
6	39.93
8	65.44
10	89.18
12	112.90
14	123.76
16	136.72

表 6.3 AISM 法の処理時間の内訳

p	T_{M_i}	$T_{C_iM_iB_i}$	T_{M_s}	T_{pre}
2	829.29	21.47	0.22	850.19
4	193.76	14.16	0.83	206.91
6	83.25	13.27	2.00	95.15
8	45.47	10.57	3.40	58.06
10	28.84	9.94	5.28	42.60
12	19.68	8.62	6.10	33.65
14	14.14	8.55	8.10	30.70
16	10.67	7.99	8.67	27.78

[数値例] 正方領域 $\Omega = [0,\ 1] \times [0,\ 1]$ における次のような偏微分方程式の境界値問題を考える．

$$\begin{aligned} -u_{xx} - u_{yy} + D\{(y - \frac{1}{2})u_x + (x - \frac{1}{3})(x - \frac{2}{3})u_y\} - 43\pi^2 u &= G(x,\ y) \\ u(x,\ y)|_{\partial\Omega} &= 1 + xy. \end{aligned}$$

ただし，右辺 $G(x,\ y)$ は真の解が $u = 1 + xy$ になるように決定する．領域 Ω を格子点数 192×192 に区切り，5 点中心差分で離散化すると得られた連立 1 次方程式の次元は $36,864$ となる．Dh の値を 2^{-7} に設定する．ただし，h はきざみ幅であり $h = 1/193$ である．

まず，グラフ分割の結果を表 6.1 に示す．第 2 列目から，pmetis はグラフをほぼ均等に分割していることが分かった．また，p の増加につれ A_i の次元が減少し，A_s の次元（シュールコンプリメントの次元）が増大する傾向が確認できる．

次に，PE 台数を $1, 2, 4, \ldots, 16$ と変えて，速度向上を計測した．ただし，部分グラフの数，つまり式 (6.1) の p の値は利用する PE 台数と一致するようにとった．速度向上を次式で定義する．

$$\frac{T_{parallel}}{T_{origin}} \tag{6.7}$$

ただし，$T_{parallel}$ は p 個の PE で 2 レベル並列計算を行い，式 (6.1) に示した P^TAP の近似逆行列を計算する時間である．T_{origin} は 1 つの PE で，AISM 法で A の近似逆行列を計算する時間である．本数値例では，$T_{origin} = 3799.65$ 秒である．計測結果を表 6.2 に示す．表 6.2 より，PE を 16 台使用すると約 136.72 倍の速度向上が得られた．また，近似逆行列の計算時間の内訳を表 6.3 に示す．各列の意味は以下のようになっている．

T_{M_i}: 行列 A_i の近似逆行列 M_i を式 (4.16) を用いて計算する時間

$T_{C_iM_iB_i}$: 近似シュールコンプリメント $\bar{S}$ のローカルの成分の計算時間

T_{M_s}: 近似シュールコンプリメント $\bar{S}$ の近似逆行列 M_s の計算時間

$T_{\mathbf{pre}}$: 前処理行列を求める時間

表 6.4 数値例 1: ILU(2) 法の一回送信するブロック数 $\tilde{m}$ と計算時間の関係

通信回数	1	2	4	8	16	32	64	128
$\tilde{m}$	2304	1152	576	288	144	72	36	18
計算時間 (秒)	0.19	0.20	0.27	0.51	0.99	1.95	4.00	7.72

各時間は各 PE の最大時間を取ったものである．これらの計算時間には，$T_{\mathrm{pre}} \approx T_{M_i} + T_{arrange} + T_{C_iM_iB_i} + T_{M_s}$ の関係がある．2,3,4 列で示した計算時間は AISM 法の総処理時間の約 99%を占める．

また，表 6.3 より，PE の数の増大につれ，行列 A_i の近似逆行列 M_i の計算時間が短縮できるものの，近似シュールコンプリメント $\bar{S}$ の近似逆行列 M_s の計算時間が長くなる．これは PE の数の増大につれ，A_i の次元が小さく，シュールコンプリメントの次元が大きくなったためであると考えられる．この事実は表 6.1 から確認できる．以上のことから，PE の台数を増やせても，前処理行列の総計算時間が必ずしも減少できるとは限らないことがわかる．

最後に，16 台の PE を用いて，3 種類の前処理行列をそれぞれ GMRES(m) 法に適用し，この偏微分方程式を離散化して得られた連立 1 次方程式を解く実験を行った．AISM 法のパラメータ s は，$s = 1.5\times \|A\|_\infty$ の値のほかに，$1.5\|A\|_2 \times 10^1$ から $1.5\|A\|_2 \times 10^2$ までの値を利用し，s を変化させたときの速度向上や残差ノルムの収束に及ぼす影響を計測した．ここで，便宜上，第 5.3 節と同じく，$\tilde{s} = s \,/\, (1.5\|A\|_\infty)$ を定義し，数値実験で s を変化させるときは $\tilde{s}$ を用いて行うものとする．また，行列 U，V の非ゼロ要素の drop off の閾値は，tolU = tolV = 0.1，もしくは 0.01 とする．

ILU 分解における一回に送信するブロック数 $\tilde{m}$ について，第 5.3 節と同様に，ブロック数 $\tilde{m}$ を変化させて ILU 分解で式 (4.1) と式 (4.2) の計算にかかる時間を計測した．その結果を表 6.4 に示す．表 6.4 から，通信回数が 1 回，つまり $\tilde{m} = 2304$ のとき計算時間が最小になったことがわかる．この事実を基に，残差ノルムの収束評価を行う数値実験では，ILU 法の一回に送信するブロック数は，$\tilde{m} = 2304$ の値を利用した．

16 台の PE を用いて，3 種類の前処理を比較する数値実験の結果を表 6.5 に示す．表 6.5 には GMRES(m) 法の残差ノルムが式 (6.6) の収束判定条件を満たすまでにかかった計算時間と反復回数を示している．この表における GMRES(m) 法の計算時間には，各前処理行列を計算するのにかかった時間も含まれており，2 列目に示されている値が各前処理行列を計算する時間である．

表 6.5 計算時間と反復回数 (time: 計算時間 (秒), iter: 反復回数)

前処理の種類	前処理行列の計算	GMERS(m) 法					
		GMRES(20)		GMRES(30)		GMRES(40)	
	time	time	iter	time	iter	time	iter
なし	0.0	–	–	–	–	–	–
AISM 法 (tolU = tolV = 0.1, $\tilde{s} = 10^0$)	27.78	–	–	1255.77	12950	1129.20	10632
AISM 法 (tolU, tolV = 0.1, $\tilde{s} = 10^1$)	38.03	2319.39	19161	1594.41	12237	1505.00	10727
AISM 法 (tolU, tolV = 0.01, $\tilde{s} = 10^0$)	114.77	926.35	1936	801.20	1637	819.35	1625
AISM 法 (tolU, tolV = 0.01, $\tilde{s} = 10^1$)	166.06	1586.82	2222	1100.54	1446	1137.44	1458
ILU(0) 法	0.51	–	–	2007.81	18258	1397.10	12164
ILU(1) 法	1.17	1352.78	9067	864.71	5650	807.39	5114
ILU(2) 法	2.17	934.05	4809	517.48	2603	605.61	2990
MR 法 (tol = 0.1, imax = 3)	556.26	–	–	–	–	–	–
MR 法 (tol = 0.1, imax = 2)	378.45	–	–	–	–	–	–
MR 法 (tol = 0.1, imax = 1)	199.37	–	–	–	–	–	–
MR 法 (tol = 0.01, imax = 3)	557.41	–	–	–	–	1277.77	16683
MR 法 (tol = 0.01, imax = 2)	377.92	–	–	–	–	–	–
MR 法 (tol = 0.01, imax = 1)	198.73	–	–	–	–	–	–

–: 最大反復回数で収束しなかった場合
imax: MR 法の反復回数

MR 法による前処理行列を適用した場合，残差ノルムを収束させることができたのは，tol = 0.01 かつ imax = 3 のときの GMRES(40) 法だけであった．ILU 分解による前処理行列を適用すると AISM 法による前処理行列を適用したときよりも少ない計算時間で前処理行列を求めることができる．しかし，AISM 法による前処理行列を適用した場合，GMRES(m) 法の反復回数は ILU 分解より少ないため，総処理時間はあまり変わらない．また，2 レベル並列計算の場合，ブロック数 $\tilde{m}$ を設定する必要がないので，ユーザの負担を軽減できる．従って，AISM 法を用いた 2 レベル並列計算は，近似逆行列の有効な並列手法の 1 つとなり得る．

6.3 本章のまとめ

本章では，部分構造法の観点に立ち，AISM 法を用いた 2 レベル並列計算を提案した．SGI 社の Origin 2400 に並列実装を行い，数値実験により，PE16 台を使用する際に，約 136 倍の速度向上が得られた．近似逆行列を並列計算で求める有効な手法の 1 つであると思われる．

しかし，本章で提案した 2 レベル並列計算を用いた並列実装では，近似シュールコンプリメントの近似逆行列を計算する部分の並列計算はしていない．p の増大，つまり利用する PE 台数の増大につれ，近似シュールコンプリメントの次元が大きくなるため，その近

似逆行列の計算時間も長くなり，前処理行列の総計算時間が必ずしも減少できるとは限らない．従って，利用する PE 台数が多ければ多いほど良いというわけではない．最大な速度向上を得るための PE の台数は問題ごとに異なる．それ以上の PE を使っても，速度向上をはかることできず，資源を十分に利用できていないという問題は残っている．この問題を解決するために，第 5 章で提案した AISM 法の部分並列実装を使って，近似シュールコンプリメントの近似逆行列を計算するのが，今後の課題である．

第 7 章

総括と結論

GMRES 法は非対称で，正則な大型疎行列を係数として持つ連立 1 次方程式を解くための有効な算法の 1 つである．しかし，クリロフ部分空間の直交基底の計算コストは，反復回数に対して線形で増加し，通常は，計算量や記憶容量などの面からリスタート版の GMRES(m) 法が用いられることが多い．GMRES 法と比べて，リスタート型の GMRES(m) 法は記憶容量が少なくてすみ，実用的な利点を持っている．一般に，GMRES(m) 法のリスタート周期 m はユーザの経験によって決められる．しかし，その残差ノルムの収束は，リスタート周期 m に大きく依存するので，問題によっては残差ノルムの収束が停滞し遅れ，適切な計算時間で収束しないこともある．

本論文では，まず，適応的なリスタートによる GMRES(m) 法の高速化について述べ，係数行列 A の近似固有値である最大 Ritz 値と最大調和 Ritz 値に注目し，適応的なリスタート手法を提案した．GMRES 法において，最大 Ritz 値と最大調和 Ritz 値は低コストで求められる．しかも，その差から GMRES(m) 法の収束が評価できる．このことから，GMRES(m) 法の中で求められる最大 Ritz 値と最大調和 Ritz 値の各々の最大値の差を利用して，リスタートするタイミングを適応的に決定することを提案した．数値実験では，固定したリスタート周期を持つ古典的な GMRES(m) 法との比較を中心に，この手法の有効性を示した．

次に，前処理としての Sherman-Morrison 法による近似逆行列 (AISM 法) の並列計算による GMRES(m) 法の高速化について考察した．AISM 法は Sherman-Morrison 公式を利用して近似逆行列を求める方法である．この手法による行列の前処理は多くの場合，優れた安定性を示している．しかし，近似逆行列の計算には，行列の列ベクトルどうしに依存関係があるので，並列化には不向きであることが指摘されている．このような難点を改善するために，本論文では，2 つの並列実装法を提案した．

まず最初に，Naik [36] の手法を用いて，AISM 法による前処理行列としての近似逆行列を求める計算手順を並列化する方法を提案した．列ベクトルを各 PE に分割し，これらのベクトルを $\tilde{m}$ 個だけ計算する度に通信を行い，計算と通信の過程を交互に繰り返すことにより部分的な並列化を可能にした．また，このような並列化により一定の速度向上が得

られた．さらに，本論文で提案した手法で計算を行った近似逆行列前処理は，MR 法を並列化して計算したものよりも残差ノルムの収束が改善できていることを確認した．従って，本論文で提案した手法は，PC クラスタシステムの環境で AISM 法による近似逆行列の計算を高速化することができるといえる．

次に，部分構造法の観点に立ち，グラフ分割を利用して係数行列 A を並列計算しやすい形式に再構成し，その近似逆行列を並列計算で求める 2 レベル並列計算に AISM 法を適用した．SGI 社の Origin 2400 に並列実装を行い，数値実験により，Naik [36] の手法を用いた部分並列実装とともに，近似逆行列を求める有効な並列処理法の 1 つとなり得ることが予想される．

謝 辞

本論文の作成に関して，数多くの方々にお世話になりました．日本での研究にわたって，野寺隆先生に大変お世話になりました．研究に関する有益な指導を始め，学会発表や論文投稿など，完備な研究環境を揃っていただきました．心よりお礼申し上げます．

また，平素より貴重なアドバイスをいただきました慶應義塾大学理工学部の谷温之先生，田村明久先生，大野義夫先生には，お忙しい中本稿を査読していただきました．この場を借りて深くお礼を申し上げます．

さらに，先輩の森屋健太郎さんと研究室の梅垣悠さん，仁科奈々さんを始め，野寺研究室の方々に様々な助力を頂き，何の支障もない子育てをしながら，研究生活を送ることができました．この場を借りて皆様に心より感謝申し上げます．

最後になりましたが，私を育てくれた両親と大学院での研究生活を支えてくれた家族にも感謝します．

参 考 文 献

[1] Arnoldi, W. E.: The Principle of Minimized Iteration in the Solution of the Matrix Eigenvalue Problem, *Quart. Appl. Math.*, Vol. 9, pp. 17–29 (1951).

[2] Anderson, E. et al.: LAPACK Users Guide, *SIAM*, Philadelpia (1992),(邦訳: 小国: 「行列演算パッケージ LAPACK 利用の手引」), 丸善 (1996)).

[3] Benzi, M., Cullum, J. K., and Tůma, M., Robust Approximate Inverse Preconditioning for the Conjugate Gradient Method, *SIAM J. Sci. Comput.*, Vol. 22, pp. 1318–1332 (2000).

[4] Benzi, M., Haws, J. C., and Tůma, M., Preconditioning Highly Indefinite and Nonsymmetric Matrices, *SIAM J. Sci. Comput.*, Vol. 22, pp. 1333–1353 (2000).

[5] Benzi, M., Meyer, C. D., and Tůma, M.: A Sparse Approximate Inverse Preconditioner for the Conjugate Gradient Method, *SIAM J. Sci. Comput.*, Vol. 17, No. 5, pp. 1135–1149 (1996).

[6] Benzi, M., and Tůma, M.: A Sparse Approximate Inverse Preconditioner for Nonsymmetric Linear Systems, *SIAM J. Sci. Comput.*, Vol.19, No. 3, pp. 968–994 (1998).

[7] Benzi, M., Marín, J., and Tůma, M.: A Two-level Parallel Preconditioner Based on Sparse Approximate Inverse, *Iterative Methods in Scienti c Computation IV, D.R. Kincaid, A. C. Elster, eds. IMACS Series in Computational and Applied Mathematics*, IMACS, New Brunswick, NJ, pp. 167–178 (1999).

[8] Bru, R., Cerdán, J., Marín, J., and Mas, J.: Preconditioning Sparse Nonsymmetric Linear Systems with the Sherman-Morrison Formula, *SIAM J. Sci. Comput.*, Vol. 25, No. 2, pp. 701–715 (2003).

[9] Burrage, K. Erhel, J.: On the Performance of Various Adaptive Preconditioned GMRES Strategies, *Numer. Linear Algebra Appl.*, Vol. 5, pp. 1010–121 (1998).

[10] Bruaset, A. M.: A Survey of Preconditioned Iterative Methods, *Pitman Research Notes in Mathematics Series*, No. 328, Longman Scientific & Technical, U. K (1995).

[11] Chow, E., and Saad, Y.: Approximate Inverse Techniques for Block-partitioned Matrices, *SIAM J. Sci. Comput.*, Vol. 18, No. 6, pp. 1657–1675 (1997).

[12] Chow, E., and Saad, Y.: Approximate Inverse Preconditioners via Sparse-sparse Iterations, *SIAM J. Sci. Comput.*, Vol. 19, No. 3, pp. 995–1023 (1998).

[13] 津野直人, 野寺隆: 適応的なリスタートを用いた ORTHOMIN(k) 法 , 情報処理学会論文誌, Vol. 40, No. 3, pp. 1350–1353 (1999).

[14] 津野直人, 野寺隆: 早期リスタートによる GMRES(m) 法の高速化 , 情報処理学会論文誌, Vol. 40, No. 4, pp. 1760–1773 (1999).

[15] Davis, T., University of Florida sparse matrix collection, NA Digest, 92(42), October 16, 1994. http://www.cise.u.edu/research/sparse/matrices. Retrieved April, 2002.

[16] Erhel, J. Burrage, K. and Pohl, B.: Restarted GMRES Preconditioned by Deflation, *J. Comp Appl. Math.*, Vol. 69, pp. 303–318 (1996).

[17] Fletcher, R.: Conjugate Gradient Methods for Indefinite Systems, *Lecture Notes in Math.*, Vol. 506, pp. 73–89 (1976).

[18] Goossens, S. and Roose, D.: Ritz and Harmonic Ritz values and the Convergence of FOM and GMRES, *Numer. Linear Algebra Appl.*, Vol. 6, pp. 281-293 (1999).

[19] Grote, M., and Huckel, T.: Parallel Preconditioning with Sparse Approximate Inverses, *SIAM J. Sci. Comput.*, Vol. 18, No. 3, pp. 838–853 (1997).

[20] Gutknecht, M. H.: Variants of BiCGStab for Matrices with Complex Spectrum, *SIAM J. Sci. Comput.*, Vol. 14, pp. 1020–1033 (1993).

[21] 羽部充，野寺隆: リスタート周期を動的に変える GMRES(m) 法, 情報処理学会論文誌, Vol. 43, No. 6, pp. 1795–1803 (2002).

[22] Hestens, M. R., and Stiefel, E.: Methods of Conjugate Gradients for Solving Linear Systems, *J. Res. Nat. Bur. Standards*, Vol. 49, pp. 409–435 (1952).

[23] Huckel, T.: Efficient Computation of Sparse Approximate Inverses, *Numerical Linear Algebra Appl.*, Vol. 5, pp. 57–71 (1998).

[24] Huckel, T.: Approximate Sparsity Patterns for the Inverse of a Matrix and Preconditioning, *Appl. Numer. Math.*, No. 30, pp. 291–303 (1999).

[25] Joubert, W.: Lanczos Methods for the Solution of Nonsymmetric Systems of Linear Equations, *SIAM J. Matrix Anal. Appl.*, Vol. 13, pp. 928–943 (1992).

[26] Karypis, G., and Kumar, V.: A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs, *SIAM J. Sci. Comput.*, Vol. 20, No. 1, pp. 359–392 (1998).

[27] Lanczos, C.: Solution of Systems of Linear Equations by Minimize Iteration, *J. Res. Nat. Bur. Standard.*, Vol. 49, pp. 33–53 (1952).

[28] Matrix Market: Collection of Test Matrices, available at:
http://math.nist.gov/MatrixMarket/index.html

[29] Morgan, R. B.: A Restarted GMRES Method Augmented with Eigenvectors, *SIAM J. Matrix Anal. Appl.*, Vol. 16, pp. 1154–1171 (1995).

[30] Morgan, R. B.: Implicitly Restarted GMRES Method and Arnoldi Methods for Nonsymmetric Systems of Equations, *SIAM J. Matrix Anal. Appl.*, Vol. 21, No. 4, pp. 1112–1135 (2000).

[31] Moriya, K. and Nodera, T.: The DEFLATED-GMRES(m, k) Method with Switching the Restart Frequency Dynamically, *Numer. Linear Algebra Appl.*, Vol. 7, pp. 569–584 (2000).

[32] 森屋健太郎, 野寺隆: 残差ノルムの収束停滞を適応的に回避する GMRES(m) 法, 情報処理学会論文誌, Vol. 43, No. 7, pp. 2264–2271 (2002).

[33] 森屋健太郎，野寺隆：並列性を考慮した大規模な線形システムの前処理，応用数理，Vol. 12, No. 1, pp. 14–28 (2002).

[34] 森屋健太郎，張臨傑，野寺隆：Sherman-Morrison 法の並列化による近似逆行列の計算，情報処理学会論文誌，Vol. 48, No. 3, pp. 1462–1478 (2007).

[35] MPICH Home Page. [Online] http://www-unix.mcs.anl.gov/mpi/mpich1/.

[36] Naik, V. K.: A Scalable Implementation of the NAS Benchmark BT on Distributed Memory Systems, *IBM Systems Journal.*, Vol. 34, No. 2, pp. 273–291 (1995).

[37] Montero, G., González, L., Flórez, E., García, M. D., and Suárez, A.: Approximate Inverse Computation Using Frobenius Inner Product, Numer. Linear Algebra Appl., Vol. 9, pp. 239–247 (2002).

[38] Nodera, T., and Noguchi, Y.: A Note on the BiCGStab(l) Method on AP1000, *(Iterative Method in Scientific Computation)* , *IMACS Series in Comp. and Applied Math.*, Vol. 4, pp. 53–58 (1998).

[39] Nodera, T., and Tsuno, N.: An Improvement of the Convergence of ORTHOMIN(k) Algorithm by Using Adaptive Restarts, *IMACS Series in Comp. and Applied Math.*, Vol. 5, pp. 99–108 (1999).

[40] 野寺隆, 野口雄一郎: AP1000 における BiCGStab(l) 法の有効性について, 情報処理学会論文誌, Vol. 38, No. 11, pp. 2089–2101 (1997).

[41] Saad, Y.: Iterative Methods for Sparse Linear Systems, PSW Publishing Co., Boston (1966).

[42] Saad, Y., and Schultz, M. H.: GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems. *SIAM J. Sci. Statist. Comput.*, Vol. 7, No. 3, pp. 856–869 (1986).

[43] Schönauer, W.: Scientific Computing on Vector Computers, North Holland, (1987).

[44] Shimoncini, Y.: On the Convergence of Restarted Krylov Subspace Method, *SIAM J. Matrix Anal. Appl.*, Vol. 22, No. 2, pp. 430–452 (2000).

[45] Sleijpen, G. L. G. and Fokkema, D. R.: BiCGStab(l) for Linear Equations Involving Unsymmetric Matrices with Complex Spectrum, *ETNA*, Vol. 1, pp. 11–32 (1993).

[46] Sleijpen, G. L. G., Van der Vorst, H. A.: A Jacobi-Davidson Iteration Method for Richardson's Method, *SIAM J. Matrix Anal. Appl.*, Vol. 17, pp. 401–425 (1996).

[47] University of Florida Sparse Matrix Collection. [Online] http://www.cise.ufl.edu/research/sparse/matrices.

[48] Van der Vorst, H. A.: The Superlinear Convergence Behaviour of GMRES, *J. Comp. Appl.*, Vol. 48, pp. 327–341, (1993).

[49] 山本哲朗: 「数値解析入門・増訂版」, サイエンス社 (2003).

[50] Zhang L., Nodera T.: A New Adaptive GMRES(m) Algorithm with Correction, *Anziam Journal.*, Vol. 46(E), pp. C409-C425 (2005).

[51] 張臨傑, 野寺隆: Ritz 値を使った GMRES 法の収束性の評価について, 情処研報, Vol. 2003, No. 102, pp. 13–18 (2003).

[52] 張臨傑, 野寺隆: Ritz 値を考慮した GMRES(m) 法の適応的なリスタート, 情報処理学会論文誌 ACS, Vol. 7, pp. 303–312 (2004).