Vol.2013-ARC-207 No.2
Vol.2013-HPC-142 No.2
2013/12/16

同期回数削減版反復解法に関する考察について

藤野清次[†2]　岩里洸介[†1]

本論文では，1 反復中の同期点を削減する手法を並列版 BiCGSafe 法と同 BiCGStar-plus 法に各々適用し，同期 1 回版の解法を導く．さらに，同期 1 回版 BiCGSafe 法，同 BiCGStar-plus 法と従来のクリロフ部分空間法について，パラメータの決定，1 反復中の同期点の個数などについて纏めて整理し，数値実験にて考案した同期 1 回版クリロフ部分空間法の並列性能を検証する．

Consideration on iterative methods with reduced number of synchronizations

Seiji Fujino [†2] and Kousuke Iwasato [†1]

In this article, we apply a technique for reduction of number of synchronization to parallel variants of BiCGSafe and BiCGStar-plus methods. Moreover we summarize characteristics of the conventional Krylov subspace method, BiCGSafe and BiCGStar-plus methods in view of determination of parameters, number of sinchronization. Through some numerical experiments, we examine parallel performance of the proposed iterative methods with single synchronization on parallel computers.

1. はじめに

大規模非対称疎行列を係数とする線形方程式 $A\boldsymbol{x} = \boldsymbol{b}$ を解くことを考える．この線形方程式の解法として積型反復法が広く用いられている．積型反復法は残差ベクトル $\boldsymbol{r} := \boldsymbol{b} - A\boldsymbol{x}$ が安定化多項式と Lanczos 多項式と初期残差ベクトルの積の形で表される解法である．積型反復法を高速化するために，解法自体の改良と並列化が重要である．

†1 九州大学大学院システム情報科学府
Graduate School of Information Science and Electrical Engineering, Kyushu University
†2 九州大学情報基盤研究開発センター Research Institute for Information Technology, Kyushu University

積型反復法の一つに BiCGSafe 法[5] がある．BiCGSafe 法では，安定化多項式のパラメータを准残差と呼ばれるベクトルの局所的な最小化に基づき決定する戦略を取り入れた．この戦略により，BiCGSafe 法は逐次計算において，従来の解法よりも安定な収束性を示すことが報告されている．さらに，准残差ベクトルの局所的な最小化の戦略と Rutishauser の交代 2 項漸化式を導入する戦略を適用した BiCGStar-plus 法[8] が 2013 年に提案され，収束の高速化が実現した．分散メモリ型コンピュータにおける並列計算において，同期による通信時間が並列数が多い場合に高速化の妨げとなる．

積型反復法において，同期点は反復中で更新される 4 種類のパラメータの更新時に行われる内積演算で発生する．特に，BiCGSafe 法と BiCGStar-plus 法では 1 反復当たり 2 回の同期点が必要となる．この 1 反復あたりの同期点を削減することにより並列計算における積型反復法の高速化が期待できる．同期点はランチョス多項式 $R_k(\lambda)$ と補助多項式 $P_k(\lambda)$ からなる交代 2 項漸化式の二つのパラメータ α_k，β_k を変形することにより削減可能である．この変形には α_k を変形する手法と β_k を変形する手法と 2 種類ある．これらの手法は准残差ベクトルの局所的な最小化の戦略を用いる解法に対して有効である．

本草稿では，1 反復中の同期点を削減する手法を BiCGSafe 法と BiCGStar-plus 法に適用し，同期 1 回版の解法を導出する．さらに，同期 1 回版 BiCGSafe 法，同 BiCGStar-plus 法と従来のクリロフ部分空間法をパラメータの決定，1 反復中の同期点の個数という観点から整理し，数値実験を用いて提案する同期 1 回版クリロフ部分空間法の並列性能を検証する．本草稿は以下のように構成される．第 2 節で，並列化のための方法について議論する．第 3 節では，二つのタイプの並列版 BiCGSafe 法を導出する．第 4 節では，二つのタイプの並列版 BiCGStar-plus 法を導出する．第 5 節では，同期 1 回版の反復解法の特徴を整理し纏める．第 6 節では，数値実験を通して，提案する並列版 BiCGSafe 法と並列版 BiCGStar-plus 法の並列性能を調べる．最後に，第 7 節で，研究報告をまとめる．ただし，時間の関係で，本草稿の数値実験には，並列版 BiCGSafe 法の結果のみしか載せられなかった．

2. 並列化

並列化には，プロセス並列とスレッド並列がある．プロセス並列化したプログラムでは，プロセスは複数のプロセッサに割り当てられ， 並列的に処理を行う．各プロセスは独立したメモリ空間を持つため， あるプロセスが他のプロセスのデータを参照する際には通信を行う必要がある．プロセス並列化には， 分散メモリ型並列計算機のみならず，共有メモリ型や階層型の並列計算機上でも利用できるという利点がある．プロセス並列化手法の標準的

Vol.2013-ARC-207 No.2
Vol.2013-HPC-142 No.2
2013/12/16

な手法として MPI(Message Passing Interface)[10] が利用される．

スレッド並列化したプログラムでは，一つのプロセスが複数個のスレッドに分けられる．スレッドは複数のプロセッサに割り当てられ，並列的に処理を実行する．全てのスレッドはメモリを共有するため，スレッド間のデータの参照が容易である．スレッド並列化の手段として，OpenMP[14] が広く用いられている．

本研究では，ノード間の並列化に MPI によるプロセス並列化を行うフラット並列化とノード間の並列化に MPI を使用し，ノード内の並列化に OpenMP を使用するハイブリット並列化の 2 種類を使用した．

3. 並列版 BiCGSafe 法

3.1 同期と内積との関係

積型反復法をプロセス並列化することを考える．積型反復法で行われる演算は，ベクトル同士の積和演算，内積演算，行列ベクトル積演算の 3 種類に分類される．このうち内積演算は積型反復法の並列計算による高速化の効果を低下させる要因の一つである．ベクトルの内積演算をプロセス並列化する場合, 積和計算は複数のプロセスに分割され, 各プロセスは個別に積和の小計を求めることになる．そのため, 各プロセスが求めた積和の小計を通信して, プロセス間で足し合わせる必要がある．プロセス並列化した場合には, 通信のために同期を取る必要があるので，この部分で通信時間の増加を招くことが多い．すなわち，1 反復中の同期点の個数が減少が並列計算における積型反復法の高速化につながると考えられる．

3.2 β_k の変形による同期 1 回版 BiCGSafe 法

BiCGSafe 法の同期点を削減し，並列計算向けの解法として改良することを考える．従来の BiCGSafe 法の 1 反復あたりの同期点はパラメータ α_k, ζ_k, η_k の更新後に 1 回とパラメータ β_k の更新後の 1 回の計 2 回である．本節では，α_k, β_k, ζ_k, η_k を一度にまとめて更新できるように β_k の更新式を変形する．β_k の定義は次のように定められる．

$$\beta_k = \frac{(\boldsymbol{r}_0^*,\ AH_kR_{k+1}\boldsymbol{r}_0)}{(\boldsymbol{r}_0^*,\ AH_kP_k\boldsymbol{r}_0)}, \tag{1}$$

式 (1) の分子は次のように変形できる．

$$(\boldsymbol{r}_0^*,\ AH_kR_{k+1}\boldsymbol{r}_0) = (A^T\boldsymbol{r}_0^*,\ H_kR_{k+1}\boldsymbol{r}_0). \tag{2}$$

したがって，式 (2) の R_{k+1} を展開すると，次の β_k の更新式が得られる．

$$\beta_k = \frac{(A^T\boldsymbol{r}_0^*,\ H_kR_k\boldsymbol{r}_0) - \alpha_k(A^T\boldsymbol{r}_0^*,\ H_kP_k\boldsymbol{r}_0)}{(\boldsymbol{r}_0^*,\ AH_kP_k\boldsymbol{r}_0)}. \tag{3}$$

式 (3) の β_k の計算には転置行列積 $A^T\boldsymbol{r}_0^*$ を必要とするが，これは反復計算前に 1 度だけ計算すればよい．また，$\boldsymbol{r}_k$ と $A\boldsymbol{p}_k$, $A^T\boldsymbol{r}_0^*$, α_k は α_k，ζ_k，η_k の計算に用いられるベクトルであり，式 (3) を用いることで，β_k を α_k，ζ_k，η_k と同時に計算することが可能となる．以上より，同期回数を削減した同期 1 回版 BiCGSafe 法が導出される．

Algorithm 1: β_k の変形による同期 1 回版 BiCGSafe 法

1. Let $\boldsymbol{x}_0$ be an initial guess, Compute $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$,
2. Choose $\boldsymbol{r}_0^*$, such that $(\boldsymbol{r}_0^*,\ \boldsymbol{r}_0) \neq 0$,
3. Compute $A^T\boldsymbol{r}_0^*$, $\boldsymbol{y}_0 = \boldsymbol{0}$, $\beta_{-1} = 0$,
4. for $k = 0, 1, \ldots$ do,
5. Compute $A\boldsymbol{r}_k$,
6. $\boldsymbol{v}_k = \boldsymbol{y}_k + \beta_{k-1}\boldsymbol{u}_{k-1}$,
7. $\boldsymbol{p}_k = \boldsymbol{r}_k + \beta_{k-1}(\boldsymbol{p}_{k-1} - \boldsymbol{u}_{k-1})$,
8. $A\boldsymbol{p}_k = A\boldsymbol{r}_k + \beta_{k-1}(A\boldsymbol{p}_{k-1} - A\boldsymbol{u}_{k-1})$,
9. if $||\boldsymbol{r}_k||/||\boldsymbol{r}_0|| \leq \epsilon$ stop,
10. $\alpha_k = \dfrac{(\boldsymbol{r}_0^*, \boldsymbol{r}_k)}{(\boldsymbol{r}^*{}_0, A\boldsymbol{p}_k)}$,
11. $\beta_k = -\dfrac{(A^T\boldsymbol{r}_0^*, \boldsymbol{r}_k) - \alpha_k(A^T\boldsymbol{r}_0^*, \boldsymbol{p}_k)}{(\boldsymbol{r}_0^*, A\boldsymbol{p}_k)}$,
12. $\zeta_k = \dfrac{(\boldsymbol{y}_k, \boldsymbol{y}_k)(A\boldsymbol{r}_k, \boldsymbol{r}_k) - (A\boldsymbol{r}_k, \boldsymbol{y}_k)(\boldsymbol{y}_k, \boldsymbol{r}_k)}{(A\boldsymbol{r}_k, A\boldsymbol{r}_k)(\boldsymbol{y}_k, \boldsymbol{y}_k) - (A\boldsymbol{r}_k, \boldsymbol{y}_k)(\boldsymbol{y}_k, A\boldsymbol{r}_k)}$,
13. $\eta_k = \dfrac{(A\boldsymbol{r}_k, A\boldsymbol{r}_k)(\boldsymbol{y}_k, \boldsymbol{r}_k) - (A\boldsymbol{r}_k, \boldsymbol{y}_k)(A\boldsymbol{r}_k, \boldsymbol{r}_k)}{(A\boldsymbol{r}_k, A\boldsymbol{r}_k)(\boldsymbol{y}_k, \boldsymbol{y}_k) - (A\boldsymbol{r}_k, \boldsymbol{y}_k)(\boldsymbol{y}_k, A\boldsymbol{r}_k)}$,
    (if $k = 0$ then $\zeta_k = \dfrac{(A\boldsymbol{r}_k, \boldsymbol{r}_k)}{(A\boldsymbol{r}_k, A\boldsymbol{r}_k)}$, $\eta_k = 0$)
14. $\boldsymbol{u}_k = \zeta_k A\boldsymbol{p}_k + \eta_k\boldsymbol{v}_k$,
15. Compute $A\boldsymbol{u}_k$,
16. $\boldsymbol{z}_k = \zeta_k\boldsymbol{r}_k + \eta_k\boldsymbol{z}_{k-1} - \alpha_k\boldsymbol{u}_k$,
17. $\boldsymbol{y}_{k+1} = \zeta_k A\boldsymbol{r}_k + \eta_k y_k - \alpha_k A\boldsymbol{u}_k$,
18. $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k\boldsymbol{p}_k + \boldsymbol{z}_k$,
19. $\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k A\boldsymbol{p}_k - \boldsymbol{y}_{k+1}$,
20. end do.

Vol.2013-ARC-207 No.2
Vol.2013-HPC-142 No.2
2013/12/16

3.3 α_k の変形による同期 1 回版 BiCGSafe 法

前項で導出した同期 1 回版 BiCGSafe 法は β_k を変形することにより，同期回数を 1 回に削減した．しかし，転置行列積 $A^T\boldsymbol{r}_0^*$ が必要であり，これが収束の安定性を低下させる可能性がある．本節では，α_k の変形により，転置行列積 $A^T\boldsymbol{r}_0^*$ を用いずに同期回数を削減する．

$$\alpha_k = \frac{(\boldsymbol{r}_0^*, \boldsymbol{r}_k)}{(\boldsymbol{r}_0^*, A\boldsymbol{p}_k)} = \frac{(\boldsymbol{r}_0^*, \boldsymbol{r}_k)}{(\boldsymbol{r}_0^*, A\boldsymbol{r}_k + \beta_k \boldsymbol{t}_{k-1})}. \tag{4}$$

パラメータ α_k に必要な内積演算 $(\boldsymbol{r}_0^*, \boldsymbol{r}_k)$，$(\boldsymbol{r}_0^*, A\boldsymbol{r}_k)$, $(\boldsymbol{r}_0^*, \boldsymbol{t}_{k-1})$ の 3 種類である．この変形により，以下のような同期 1 回版 BiCGSafe 法のアルゴリズムが導出される．

Algorithm 2: α_k の変形による同期 1 回版 BiCGSafe 法

1. Let $\boldsymbol{x}_0$ be an initial guess, Compute $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$,
2. Choose $\boldsymbol{r}_0^*$ such that $(\boldsymbol{r}_0^*, \boldsymbol{r}_0) \neq 0$, $\beta_{-1} = 0$,
3. **for** $k = 0, 1, \ldots$ **do**,
4. Compute $A\boldsymbol{r}_k$,
5. **if** $||\boldsymbol{r}_k||/||\boldsymbol{r}_0|| \leq \epsilon$ **stop**,
6. $\beta_k = \dfrac{\alpha_{k-1}}{\zeta_{k-1}} \dfrac{(\boldsymbol{r}_0^*, \boldsymbol{r}_k)}{(\boldsymbol{r}_0^*, \boldsymbol{r}_{k-1})}$,
7. $\alpha_k = \dfrac{(\boldsymbol{r}_0^*, \boldsymbol{r}_k)}{(\boldsymbol{r}_0^*, A\boldsymbol{r}_k) + \beta_k(\boldsymbol{r}_0^*, \boldsymbol{t}_{k-1})}$,
8. $\zeta_k = \dfrac{(\boldsymbol{y}_k, \boldsymbol{y}_k)(A\boldsymbol{r}_k, \boldsymbol{r}_k) - (A\boldsymbol{r}_k, \boldsymbol{y}_k)(\boldsymbol{y}_k, \boldsymbol{r}_k)}{(A\boldsymbol{r}_k, A\boldsymbol{r}_k)(\boldsymbol{y}_k, \boldsymbol{y}_k) - (A\boldsymbol{r}_k, \boldsymbol{y}_k)(\boldsymbol{y}_k, A\boldsymbol{r}_k)}$,
9. $\eta_k = \dfrac{(A\boldsymbol{r}_k, A\boldsymbol{r}_k)(\boldsymbol{y}_k, \boldsymbol{r}_k) - (A\boldsymbol{r}_k, \boldsymbol{y}_k)(A\boldsymbol{r}_k, \boldsymbol{r}_k)}{(A\boldsymbol{r}_k, A\boldsymbol{r}_k)(\boldsymbol{y}_k, \boldsymbol{y}_k) - (A\boldsymbol{r}_k, \boldsymbol{y}_k)(\boldsymbol{y}_k, A\boldsymbol{r}_k)}$,
   (**if** $k = 0$ **then** $\alpha_k = \dfrac{(\boldsymbol{r}_0^*, \boldsymbol{r}_k)}{(\boldsymbol{r}_0^*, A\boldsymbol{r}_k)}$, $\beta_k = 0$, $\zeta_k = \dfrac{(A\boldsymbol{r}_k, \boldsymbol{r}_k)}{(A\boldsymbol{r}_k, A\boldsymbol{r}_k)}$, $\eta_k = 0$),
10. $\boldsymbol{p}_k = \boldsymbol{r}_k + \beta_k(\boldsymbol{p}_{k-1} - \boldsymbol{u}_{k-1})$,
11. $A\boldsymbol{p}_k = A\boldsymbol{r}_k + \beta_k \boldsymbol{t}_{k-1}$,
12. $\boldsymbol{u}_k = \zeta_k A\boldsymbol{p}_k + \eta_k(\boldsymbol{y}_k + \beta_k \boldsymbol{u}_{k-1})$,
13. Compute $A\boldsymbol{u}_k$,
14. $\boldsymbol{t}_k = A\boldsymbol{p}_k - A\boldsymbol{u}_k$,
15. $\boldsymbol{z}_k = \zeta_k \boldsymbol{r}_k + \eta_k \boldsymbol{z}_{k-1} - \alpha_k \boldsymbol{u}_k$,
16. $\boldsymbol{y}_{k+1} = \zeta_k A\boldsymbol{r}_k + \eta_k \boldsymbol{y}_k - \alpha_k A\boldsymbol{u}_k$,
17. $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k + \boldsymbol{z}_k$,
18. $\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k A\boldsymbol{p}_k - \boldsymbol{y}_{k+1}$,
19. **end do**.

4. 並列版 BiCGStar-plus 法

4.1 β_k の変形による同期 1 回版 BiCGStar-plus 法

3.2 節の同期点を削減する手法は BiCGStar-plus 法にも適用できる．以下に β_k を変形した同期 1 回版 BiCGStar 法のアルゴリズムを示す．

Algorithm 3: β_k の変形による同期 1 回版 BiCGStar-plus 法

1. Let $\boldsymbol{x}_0$ be an initial guess, Compute $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$, choose $\tilde{\boldsymbol{r}}_0$,
2. Compute $A\boldsymbol{r}_0$, $A^T\tilde{\boldsymbol{r}}_0$, $\boldsymbol{p}_0 = \boldsymbol{r}_0$, $A\boldsymbol{p}_0 = A\boldsymbol{r}_0$, $\boldsymbol{y}_0 = \boldsymbol{s}_0 = \boldsymbol{t}_0 = \boldsymbol{0}$,
3. **for** $k = 0, 1, \ldots$ **do**,
4. **if** $||\boldsymbol{r}_k||/||\boldsymbol{r}_0|| \leq \epsilon$ **stop**,
5. $\alpha_k = \dfrac{(\tilde{\boldsymbol{r}}_0, \boldsymbol{r}_k)}{(\tilde{\boldsymbol{r}}_0, A\boldsymbol{p}_k)}$,
6. $\beta_k = \dfrac{(A^T\tilde{\boldsymbol{r}}_0, \boldsymbol{r}_k) - \alpha_k(A^T\tilde{\boldsymbol{r}}_0, A\boldsymbol{p}_k)}{(\tilde{\boldsymbol{r}}_0, A\boldsymbol{p}_k)}$,
7. $\zeta_k = \dfrac{(\boldsymbol{y}_k, \boldsymbol{y}_k)(A\boldsymbol{r}_k, \boldsymbol{r}_k) - (A\boldsymbol{r}_k, \boldsymbol{y}_k)(\boldsymbol{y}_k, \boldsymbol{r}_k)}{(A\boldsymbol{r}_k, A\boldsymbol{r}_k)(\boldsymbol{y}_k, \boldsymbol{y}_k) - (A\boldsymbol{r}_k, \boldsymbol{y}_k)(\boldsymbol{y}_k, A\boldsymbol{r}_k)}$,
8. $\eta_k = \dfrac{(A\boldsymbol{r}_k, A\boldsymbol{r}_k)(\boldsymbol{y}_k, \boldsymbol{r}_k) - (A\boldsymbol{r}_k, \boldsymbol{y}_k)(A\boldsymbol{r}_k, \boldsymbol{r}_k)}{(A\boldsymbol{r}_k, A\boldsymbol{r}_k)(\boldsymbol{y}_k, \boldsymbol{y}_k) - (A\boldsymbol{r}_k, \boldsymbol{y}_k)(\boldsymbol{y}_k, A\boldsymbol{r}_k)}$,
   (**if** $k = 0$ **then** $\zeta_k = \dfrac{(A\boldsymbol{r}_k, \boldsymbol{r}_k)}{(A\boldsymbol{r}_k, A\boldsymbol{r}_k)}$, $\eta_k = 0$),
9. $\boldsymbol{v}_k = \zeta_k \boldsymbol{r}_k + \eta_k \boldsymbol{t}_k$,
10. $\boldsymbol{z}_k = \zeta_k A\boldsymbol{r}_k + \eta_k \boldsymbol{y}_k$,
11. $\boldsymbol{c}_k = \zeta_k A\boldsymbol{p}_k + \eta_k \boldsymbol{s}_k$,
12. Compute $A\boldsymbol{c}_k$,
13. $\boldsymbol{w}_k = \boldsymbol{p}_k - \boldsymbol{c}_k$,
14. $A\boldsymbol{w}_k = A\boldsymbol{p}_k - A\boldsymbol{c}_k$,
15. $\boldsymbol{t}_{k+1} = \boldsymbol{v}_k - \alpha_k \boldsymbol{c}_k$,
16. $\boldsymbol{y}_{k+1} = \boldsymbol{z}_k - \alpha_k A\boldsymbol{c}_k$,
17. $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{v}_k + \alpha_k \boldsymbol{w}_k$,
18. $\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \boldsymbol{z}_k - \alpha_k A\boldsymbol{w}_k$,
19. Compute $A\boldsymbol{r}_{k+1}$,
20. $\boldsymbol{s}_{k+1} = \boldsymbol{y}_{k+1} - \beta_k \boldsymbol{c}_k$,
21. $\boldsymbol{p}_{k+1} = \boldsymbol{r}_{k+1} - \beta_k \boldsymbol{w}_k$,
22. $A\boldsymbol{p}_{k+1} = A\boldsymbol{r}_{k+1} - \beta_k A\boldsymbol{w}_k$,
23. **end do**.

Vol.2013-ARC-207 No.2
Vol.2013-HPC-142 No.2
2013/12/16

4.2 α_k の変形による同期 1 回版 BiCGStar-plus 法

BiCGSafe 法に適用された，α_k を変形することによる同期回数削減手法は BiCGStar-plus 法にも適用できる．α_k，β_k の計算は，

$$\beta_k = -\frac{\alpha_{k-1}}{\zeta_{k-1}} \frac{(\tilde{\boldsymbol{r}}_0, H_k(A)R_k(A)\boldsymbol{r}_0)}{(\tilde{\boldsymbol{r}}_0, H_{k-1}(A)R_{k-1}(A)\boldsymbol{r}_0)} \tag{5}$$

$$\begin{aligned}\alpha_k &= \frac{(\tilde{\boldsymbol{r}}_0, H_k(A)R_k(A)\boldsymbol{r}_0)}{(\tilde{\boldsymbol{r}}_0, AH_k(A)P_k(A)\boldsymbol{r}_0)} = \frac{(\tilde{\boldsymbol{r}}_0, H_k(A)R_k(A)\boldsymbol{r}_0)}{(\tilde{\boldsymbol{r}}_0, AH_k(A)R_k(A)\boldsymbol{r}_0 - \beta_k AH_k(A)P_{k-1}(A)\boldsymbol{r}_0)} \\ &= \frac{(\tilde{\boldsymbol{r}}_0, H_k(A)R_k(A)\boldsymbol{r}_0)}{(\tilde{\boldsymbol{r}}_0, AH_k(A)R_k(A)\boldsymbol{r}_0) - (\tilde{\boldsymbol{r}}_0, \beta_k AH_k(A)P_{k-1}(A)\boldsymbol{r}_0)}\end{aligned} \tag{6}$$

と表される．以下に，α_k 変形版 BiCGStar-plus 法の算法を示す．

Algorithm 4: α_k の変形による同期 1 回版 BiCGStar-plus 法

1. Let $\boldsymbol{x}_0$ be an initial guess, Compute $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$, choose $\tilde{\boldsymbol{r}}_0$,
2. **for** $k = 0, 1, \ldots$ **do**,
3. Compute $A\boldsymbol{r}_k$,
4. **if** $||\boldsymbol{r}_k||/||\boldsymbol{r}_0|| \leq \epsilon$ **stop**,
5. $\beta_k = -\frac{\alpha_{k-1}}{\zeta_{k-1}} \frac{(\tilde{\boldsymbol{r}}_0, \boldsymbol{r}_k)}{(\tilde{\boldsymbol{r}}_0, \boldsymbol{r}_{k-1})}$,
6. $\alpha_k = \frac{(\tilde{\boldsymbol{r}}_0, \boldsymbol{r}_k)}{(\tilde{\boldsymbol{r}}_0, A\boldsymbol{r}_k) - \beta_k(\tilde{\boldsymbol{r}}_0, A\boldsymbol{w}_{k-1})}$,
7. $\zeta_k = \frac{(\boldsymbol{y}_k, \boldsymbol{y}_k)(A\boldsymbol{r}_k, \boldsymbol{r}_k) - (A\boldsymbol{r}_k, \boldsymbol{y}_k)(\boldsymbol{y}_k, \boldsymbol{r}_k)}{(A\boldsymbol{r}_k, A\boldsymbol{r}_k)(\boldsymbol{y}_k, \boldsymbol{y}_k) - (A\boldsymbol{r}_k, \boldsymbol{y}_k)(\boldsymbol{y}_k, A\boldsymbol{r}_k)}$,
8. $\eta_k = \frac{(A\boldsymbol{r}_k, A\boldsymbol{r}_k)(\boldsymbol{y}_k, \boldsymbol{r}_k) - (A\boldsymbol{r}_k, \boldsymbol{y}_k)(A\boldsymbol{r}_k, \boldsymbol{r}_k)}{(A\boldsymbol{r}_k, A\boldsymbol{r}_k)(\boldsymbol{y}_k, \boldsymbol{y}_k) - (A\boldsymbol{r}_k, \boldsymbol{y}_k)(\boldsymbol{y}_k, A\boldsymbol{r}_k)}$,
   (**if** $k = 0$ **then** $\alpha_k = \frac{(\tilde{\boldsymbol{r}}_0, \boldsymbol{r}_k)}{(\tilde{\boldsymbol{r}}_0, A\boldsymbol{r}_k)}$, $\beta_k = 0$, $\zeta_k = \frac{(A\boldsymbol{r}_k, \boldsymbol{r}_k)}{(A\boldsymbol{r}_k, A\boldsymbol{r}_k)}$, $\eta_k = 0$),
9. $\boldsymbol{s}_k = \boldsymbol{y}_k - \beta_k \boldsymbol{c}_{k-1}$,
10. $\boldsymbol{p}_k = \boldsymbol{r}_k - \beta_k \boldsymbol{w}_{k-1}$,
11. $A\boldsymbol{p}_k = A\boldsymbol{r}_k - \beta_k A\boldsymbol{w}_{k-1}$,
12. $\boldsymbol{v}_k = \zeta_k \boldsymbol{r}_k + \eta_k \boldsymbol{t}_k$,
13. $\boldsymbol{z}_k = \zeta_k A\boldsymbol{r}_k + \eta_k \boldsymbol{y}_k$,
14. $\boldsymbol{c}_k = \zeta_k A\boldsymbol{p}_k + \eta_k \boldsymbol{s}_k$,
15. Compute $A\boldsymbol{c}_k$,
16. $\boldsymbol{w}_k = \boldsymbol{p}_k - \boldsymbol{c}_k$,
17. $A\boldsymbol{w}_k = A\boldsymbol{p}_k - A\boldsymbol{c}_k$,
18. $\boldsymbol{t}_{k+1} = \boldsymbol{v}_k - \alpha_k \boldsymbol{c}_k$,
19. $\boldsymbol{y}_{k+1} = \boldsymbol{z}_k - \alpha_k A\boldsymbol{c}_k$,
20. $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{v}_k + \alpha_k \boldsymbol{w}_k$,
21. $\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \boldsymbol{z}_k - \alpha_k A\boldsymbol{w}_k$,
22. **end do**.

5. 同期 1 回版の反復法のまとめ

本節では，各解法の算法上の比較を行う．表 1 にパラメータ α_k, β_k の比較を示す．BiCGSafe 法は η_k, ζ_k の更新に准残差ベクトルを使用することにより 1 反復あたりの同期点が 2 回に削減された．β_k 変形版の解法では β_k の更新に A^T が必要となる．一方，α_k 変形版の解法では A^T を使用せずに α_k, β_k を更新できる．

表 1 二つのパラメータの更新式と同期点個数などの特徴．

解法の種類	パラメータの更新式		特　徴
	α_k	β_k	
GPBiCG	$\frac{(\boldsymbol{r}_0^*, \boldsymbol{r}_k)}{(\boldsymbol{r}_0^*, A\boldsymbol{p}_k)}$	$\frac{\alpha_k}{\zeta_k} \cdot \frac{(\boldsymbol{r}_0^*, \boldsymbol{r}_{k+1})}{(\boldsymbol{r}_0^*, \boldsymbol{r}_k)}$	α_k と β_k 個別更新． η_k, ζ_k 同 時更新．同期 3 回
Safe 法 Star-plus	同上	同上	α_k, η_k, ζ_k の更新時 と β_k の更新時の 2 回同期必要
β_k 変形	同上	$-\frac{(A^T\boldsymbol{r}_0^*, \boldsymbol{r}_k) - \alpha_k(A^T\boldsymbol{r}_0^*, \boldsymbol{p}_k)}{(\boldsymbol{r}_0^*, A\boldsymbol{p}_k)}$	α_k, β_k, η_k, ζ_k を 1 度に更新． 同期 1 回．A^T 必要
α_k 変形	$\frac{(\boldsymbol{r}_0^*, \boldsymbol{r}_k)}{(\boldsymbol{r}_0^*, A\boldsymbol{r}_k + \beta_k \boldsymbol{t}_{k-1})}$	GPBiCG 法と同じ	β_k 変形版と同じく同期 1 回． A^T は不要

次に，表 2 に BiCGSafe 法と BiCGStar-plus 法の 1 反復当たりの同期回数と安定化多項式の更新する漸化式を示す．表中の β_k, α_k はそれぞれ β_k 変形版，α_k 変形版の解法を示す．BiCGSafe 法は安定化多項式の更新に交代 2 項漸化式を用いる．一方，BiCGStar-plus 法は Rutishauser の交代 2 項漸化式を用いる．また，両解法とも原型版の 1 反復当たりの同期回数は 2 回である．

5.1 フラット並列化とハイブリット並列化

BiCGSafe 法と BiCGStar-plus 法の並列化はフラット並列化とハイブリット並列化の 2 種類で行った．フラット並列では，全てのプロセッサでプロセスを実行するため，並列化性能に対するプロセス間通信の影響が大きく，効果的な並列化が行えない可能性がある．ハイブリッド並列化ではノード内ではスレッドによる並列処理が可能であり，フラット並列化と

表 2 二つの解法の同期点数と漸化式のまとめ.

	GPBiCG 法	BiCGSafe 法			BiCGStar-plus 法		
		原型	β_k 変形	α_k 変形	原型	β_k 変形	α_k 変形
同期点数	3 回	2 回	1 回	1 回	2 回	1 回	1 回
漸化式	通常の交代 2 項漸化式				Rutishauser*の交代 2 項漸化式		

比べてプロセス間の通信量を削減できるという利点がある.

6. 数 値 実 験

すべての演算は倍精度浮動小数点演算で行った．使用した計算機は CX400 (Intel Xeon E5-2690, clock of 2.93GHz, 128GB, OS: Red Hat Linux Enterprise) である．Fortran90 の最適化オプションは “-Kfast” を使用した．収束判定条件は $||\boldsymbol{r}_{k+1}||_2/||\boldsymbol{r}_0||_2 \leq 10^{-8}$. 反復の初期値はすべて零とした時間の計測は各々のケースについて合計 5 回行い，最大と最小の場合を除く 3 回の平均値を採用した．プロセス数は 2, 4, 8, 16, 32 まで，各プロセスでは 8 スレッドを使用した.

6.1 テスト行列

表 3 にテスト行列[13] の主な特徴示す．表中の “nnz” は行列の総非零要素数，“ave_nnz” は 1 行当りの平均総非零要素数を各々表す.

表 3 テスト行列の主な特徴.

matrix	dimension	nnz	ave_nnz
epb3	84,617	463,625	5.5
tmt_unsym	917,825	4,584,801	5.0
xenon2	157,464	3,866,688	24.6

6.2 数値実験結果

表 4 に 3 種類の行列に対する並列性能を示す．調べた並列版反復法は以下の 4 種類である.

(1) GPBiCG 法
(2) BiCGSafe 法
(3) BiCGSafe-β 法
(4) BiCGSafe-α 法

ここで，二つの同期 1 回版 BiCGSafe 法のうち，α_k を変形した解法を BiCGSafe-α 法，同様に β_k を変形した解法を BiCGSafe-β 法と各々表記する.

表中の台数効果 (speed up) とは，1 プロセスでの計算速度を 1 としたときに他のプロセス数における計算速度が何倍速くなったかを表す．また，TRR(True Relative Residual) とは，収束して得られた近似解 $\boldsymbol{x}_{k+1}$ に対して $(\log_{10}(||\boldsymbol{b}-A\boldsymbol{x}_{k+1}||_2/||\boldsymbol{b}-A\boldsymbol{x}_0||_2))$ で表される値である．また，“nPE” とは PE 数 (number of Processing Element) の略である．さらに，“$M\boldsymbol{v}$” とは，行列ベクトルの積の回数を表す．“tot.time” とは，合計経過時間，“ave.time” とは，反復 1 回当りの平均時間を各々表す．合計経過時間の欄の太字の数字は各行列で最短の時間を表す.

表 4 3 種類の行列に対する並列性能.
(a)matrix: epb3

method	nPE	$M\boldsymbol{v}$	tot.time [sec.]	ratio	ave.time [msec.]	speed up	TRR
GPBiCG	1	3,852	7.083	1.00	1.839	1.00	-8.0
	16	4,056	0.509	1.00	0.125	14.65	-8.0
	64	4,178	0.211	1.00	0.051	36.41	-8.0
	256	4,114	0.174	1.00	0.042	43.48	-8.0
BiCGSafe	1	3,764	6.172	0.87	1.640	1.00	-8.0
	16	3,770	0.415	0.82	0.110	14.90	-8.0
	64	3,956	0.179	0.85	0.045	36.24	-8.0
	256	3,684	0.130	0.75	0.035	46.47	-8.0
BiCGSafe-β	1	3,910	6.632	0.94	1.696	1.00	-8.0
	16	3,870	0.427	0.84	0.110	15.37	-8.0
	64	3,858	0.163	0.77	0.042	40.15	-8.0
	256	3,844	0.106	0.61	0.028	61.51	-8.0
BiCGSafe-α	1	3,716	6.320	0.89	1.701	1.00	-8.0
	16	3,708	0.410	0.81	0.111	15.38	-8.0
	64	3,518	0.143	0.68	0.041	41.84	-8.0
	256	3,722	**0.101**	0.58	0.027	62.68	-8.0

表 4 に示した結果から，同期 1 回版の BiCGSafe-β 法と同 BiCGSafe-α 法の並列性能のよさがわかる.

7. お わ り に

本論文では，1 反復中の同期点を削減する手法を並列版 BiCGSafe 法と同 BiCGStar-plus 法に各々適用し，同期 1 回版の解法を導いた．さらに，同期 1 回版 BiCGSafe 法，同 BiCGStar-plus 法と従来のクリロフ部分空間法について，パラメータの決定，1 反復中の同期点の個数などについて纏めた．数値実験にて，新しい同期 1 回版クリロフ部分空間法の並

(b)matrix: tmt_unsym

method	nPE	$M\boldsymbol{v}$	tot.time [sec.]	ratio	ave.time [msec.]	speed up	TRR
GPBiCG	1	8,900	207.092	1.00	23.269	1.00	-6.7
	16	8,866	25.308	1.00	2.855	8.15	-5.8
	64	9,146	3.421	1.00	0.374	62.21	-5.6
	256	11,600	1.665	1.00	0.144	162.11	-2.9
BiCGSafe	1	10,054	219.519	1.06	21.834	1.00	-7.5
	16	9,128	22.314	0.88	2.445	8.93	-7.6
	64	9,662	3.080	0.90	0.319	68.49	-6.4
	256	8,820	1.070	0.64	0.121	179.98	-7.6
BiCGSafe-β	1	8,718	193.999	0.94	22.253	1.00	-7.2
	16	9,080	23.580	0.93	2.597	8.57	-7.6
	64	8,588	3.283	0.96	0.382	58.21	-7.6
	256	9,122	**1.067**	0.64	0.117	190.24	-7.2
BiCGSafe-α	1	10,572	224.783	1.09	21.262	1.00	-7.6
	16	9,024	22.663	0.90	2.511	8.47	-7.2
	64	8,848	2.821	0.82	0.319	66.69	-7.6
	256	10,184	1.230	0.74	0.121	176.04	-7.6

(c)matrix: xenon2

method	nPE	$M\boldsymbol{v}$	tot.time [sec.]	ratio	ave.time [msec.]	speed up	TRR
GPBiCG	1	1,496	12.405	1.00	8.292	1.00	-8.0
	16	1,610	1.423	1.00	0.884	9.38	-8.0
	64	1,764	0.285	1.00	0.162	51.32	-8.0
	256	1,758	0.143	1.00	0.081	101.94	-8.0
BiCGSafe	1	1,364	10.847	0.87	7.952	1.00	-8.0
	16	1,524	1.260	0.89	0.827	9.62	-8.0
	64	1,410	0.216	0.76	0.153	51.91	-8.0
	256	1,634	0.122	0.85	0.075	106.51	-8.0
BiCGSafe-β	1	1,762	14.005	1.13	7.948	1.00	-8.0
	16	1,762	1.504	1.06	0.854	9.31	-8.0
	64	1,660	0.249	0.87	0.150	52.99	-8.0
	256	1,718	0.113	0.79	0.066	120.84	-8.0
BiCGSafe-α	1	1,520	12.436	1.00	8.182	1.00	-8.0
	16	1,684	1.405	0.99	0.834	9.81	-8.0
	64	1,450	0.215	0.75	0.148	55.18	-8.0
	256	1,436	**0.099**	0.69	0.069	118.67	-8.0

列性能を検証した．その結果，同期 1 回版の BiCGSafe-β 法と同 BiCGSafe-α 法の並列性能が優れていることがわかった．

参 考 文 献

1) Abe, K., Sleijpen, Gerard L.G.: Solving linear equations with a stabilized GPBiCG method, Appl. Numer. Math., doi:10.1016/j.apnum.2011.06.10, (2011).
2) 阿部 邦美，曽我部 知広，藤野 清次，張 紹良：非対称行列用共役残差法に基づく積型反復解法，情報処理学会論文誌，Vol.48，No.SIG8，pp.11–21, (2007).
3) 馬場慎也: プロセス並列化した PAGME つき CG 法向きの効率的な通信方法の提案, 九州大学大学院システム情報科学府情報工学専攻修士論文，(2009)
4) Chow, E., Hysom, D.: Assessing Performance of Hybrid MPI/OpenMP Programs on SMP Clusters, Technical Report UCRL-JC-143957, Lawrence Livermore National Laboratory, (2001).
5) Fujino, S., Fujiwara, M., Yoshida, M.: A proposal of preconditioned BiCGSafe method with safe convergence, Proc. of The 17th IMACS World Congress on Scientific Computation, Appl. Math. Simul., CD-ROM, Paris, France, (2005).
6) 藤原 牧: 非対称行列に対する不完全分解前処理と反復法の評価，九州大学大学院システム情報科学府情報工学専攻，修士論文，(2007).
7) 草場健一郎: 並列版疎行列ベクトル積における負荷分散最適化手法, 九州大学大学院システム情報科学府情報工学専攻, (2010)
8) Murakami, K., A proposal of a product type iterative method using associate residual for parallel computer, Proc. of International workshop on HPC, Krylov Subspace method and its applications, pp.23-26, (2013).
9) 村上啓一: 同期点を削減した並列計算向き Krylov 部分空間法の提案, 九州大学大学院システム情報科学府修士論文 (2013)
10) Pacheco, P.: Parallel Programming with MPI, Morgan Kaufmann Publishers, (1997), 秋葉博訳: MPI 並列プログラミング, 培風館, 東京, (2001).
11) Rutishauser, H.: Theory of gradient method, Refined Iterative Methods for Computation of the Solution and the Eigenvalues of Self-Adjoint Boundary Value Problems, Mitt. Inst. Angew. Math. ETH Zürich, Birkhäuser, Basel, pp.24-49, (1959).
12) Silva, M., Wait, R.: Sparse Matrix Storage Revisited, Proceedings of the 2nd conference on Computing frontiers, pp.230-235, 2005.
13) University of Florida Sparse Matrix Collection: http://www.cise.ufl.edu/research/sparse/matrices/index.html.
14) 牛島省: OpenMP による並列プログラミングと数値計算法, 丸善, (2006).