Vol.2009-HPC-121 No.5
2009/8/4

線形方程式求解に向けた前処理付き クリロフ部分空間法と収束判定に関する考察

伊藤祥司[†]　　杉原正顯[††]

線形方程式求解アルゴリズムの体系的な性能比較のフレームワークを紹介し，そこで確認されたスケーリングも含めた前処理方式に関わる偽収束（見かけ上の収束）の原因について議論する．そして，典型的な前処理方式と各々に対する収束判定，および，残差ベクトルとの密接な関係について議論し，対称系に対する共役勾配法の前処理は両側以外にも左右方向も存在することと，非対称系の左前処理アルゴリズムは２系統存在することも示す．

A study of a relation between preconditioned Krylov subspace methods and their convergence criterions to solve linear equations

Shoji Itoh[†]　and　Masaaki Sugihara[††]

A framework of systematical performance evaluation for comparing numerical algorithms of linear equations is introduced and some cause of the false convergence brought by preconditioning way and scaling are discussed. In this paper, especially, the close relations of preconditioning way, its convergence criterion and definition of residual vector are discussed. Further, the existence of the right or left side preconditioning for the conjugate gradient method despite of symmetric system and the existence of the two-types of left preconditioning system for non-symmetric system are shown.

1. はじめに

自然現象や工学現象の解明では，数値シミュレーションを用いた解析が盛んである．その過程では，多くの場合，大規模な $n \times n$ の係数行列を持つ線形方程式

$$Ax = b \tag{1}$$

の求解に帰着され，シミュレーションに要する時間の多くがこの計算に費やされる．

ところが，線形方程式の求解アルゴリズムには様々なものが存在し，対象とする問題の性質によっては，その性能が十分に発揮されないような場合や数値解が得られない場合もある[1]．更には，アルゴリズムの実効性能などに関しては，理論面だけでは分析し切れないことも少なく無い．これまでに，このような観点から求解アルゴリズムに対する体系的な性能評価と特性分析を行い，そこで得られた数値実験結果のデータ分析をとおして新しい知見を見出してきている[2,3]．

また，近年の国家プロジェクトでもある次世代スーパーコンピュータ開発などでは，超並列計算により大規模な問題の求解が行なわれる．しかし，そのような状況においても，逐次計算にも共通した，求解アルゴリズム面での安定求解に関係する問題点が潜む．並列計算の段階でそれらの問題点に直面した場合，そこでの問題解決は逐次計算と比較して遥かに複雑である．従って，逐次計算に潜在している“求解品質”を大きく揺さぶる問題点の発見と解決が重要である．その一手段としても，品質管理手法に基づいた体系的な性能評価や特性分析の研究が展開されている[4]．

本稿では，前処理を施した系と収束判定との関係に着目して，実験を通して確認された諸問題点を指摘し，数値シミュレーションの安定求解に向けて考察した．特にスケーリングの思わぬ影響を確認したので，それについても説明する[3]．そして，本稿では新たに，従来から十分に認識されているとは言い難い前処理系と収束判定との関係について説明し，対称系に対する共役勾配法の前処理は両側以外にも左右方向のものも存在することと，非対称系の左前処理アルゴリズムは２系統存在することも示す．

2. 性能情報 DB の構築環境について

本研究では，数値計算ライブラリ Lis（Library of Iterative Solvers for Linear Systems）[5] の逐次コードを使用した．バージョンは lis-1.1.2 である．本稿の中では，Krylov

[†] 理化学研究所 情報基盤センター
Advanced Center for Computing and Communication, RIKEN
[††] 東京大学大学院 情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo

部分空間法に基づく反復解法 9 種類（CG, BiCG, CGS, BiCGStab, BiCGStab(*l*), GPBiCG, TFQMR, Orthomin, GMRES)，前処理 10 種類（前処理無し，点 Jacobi, ILU, SSOR, Hybrid, I+S, SAINV, SAAMG, Crout ILU, ILUT）による結果を示す．Lis で用意されているアルゴリズム中で用いられるパラメタ値は，SAAMG のみ “-saamg_unsym true” を指定し，それ以外は Lis のデフォルト値を用いた．

線形方程式のテスト問題では，Matrix Market[6]の中から線形方程式向きのテスト行列を用いて問題を用意した．右辺項のベクトルは，解ベクトルの全要素を 1.0 とし(1)式に代入して生成した．各アルゴリズムに与える初期ベクトルには，零ベクトルを用いた．収束判定は，アルゴリズム中の残差ベクトルに対する相対残差ノルムが

$$\|r_k\|_2 / \|b\|_2 \le 1.0 \times 10^{-12} \qquad (2)$$

を満たすときとした（r_k はアルゴリズム中の残差ベクトル，k は反復回数）．最大反復回数には行列のサイズを用いた．これらの条件は，新しい数値計算アルゴリズムを提案した際の数値実験でも採用される典型的なテスト問題の内の一つである．

これらを用いてジョブを実行し，データ採取するために構築した情報システムとして，計算サーバに HP ProLiant DL585 G2（CPU: AMD Opteron 8220, 2.8GHz x86_64，メモリサイズ: 64GB，OS: RedHat Linux 2.6.9-42.ELsmp，コンパイラ: Intel icc 10.1, ifort 10.1）を用いた．ここでは，アルゴリズムの数理的特性を評価するために，1CPU，1 コアで計算した．コンパイルオプションは，Lis の Makefile に記載されている値を用いた．

以上から，全線形方程式に対して Lis の全アルゴリズムを用いて求解したとき，全て同一条件の計算環境における計算実験を実施でき、そこでの計算結果を性能情報 DB に蓄積した．ここで特に重要なことは，計算サーバの環境，すなわち，プロセッサ，メモリサイズ，コンパイラなどを常に同じ条件にした上でジョブ実行し，データ採取することである．

3. 線形方程式求解アルゴリズムにおける前処理

線形方程式を反復法により解く場合，解法と併用する前処理には，大きく次のものが挙げられる．

■ 前処理：狭義の意味での前処理であるが，一般に反復法に対する前処理と言う場合，こちらを指す．

$K \approx A$ となる行列 K を作り，$K = K_L K_R$ と分解し，

$$K_L^{-1} A K_R^{-1} K_R x = K_L^{-1} b \qquad (3)$$

と式(1)を変換する．このような前処理を施すことにより，係数行列が単位行列に近くなり，条件数の改善と固有値分布が 1.0 付近に密集するため，反復法の収束性が向上することが期待される．ただし，実際の求解では，(3)自体は実施せず，方程式(3)の求解と同値となる前処理付きアルゴリズムにより解く．従って，求解アルゴリズムに入力する情報（係数行列，右辺項）は，(1)で用いられている情報のままである．

式(3)は，線形方程式(1)の両側から前処理を施したものであり両側前処理と呼ばれる．式(3)で $K_L = I, K_R = K$ としたものを右前処理，$K_L = K, K_R = I$ としたものを左前処理と呼び，各々は，

$$AK^{-1}Kx = b, \qquad (4)$$

$$K^{-1}Ax = K^{-1}b \qquad (5)$$

と表される．

■ スケーリング：係数行列の値の大きさを揃えることにより，数値計算上の安定性を向上させる．

その演算方法は，元の係数行列の対角要素 D=diag(A)の逆数から構成される行列を施す方法

$$D^{-1}Ax = D^{-1}b. \qquad (6)$$

の他に，元の係数行列の対角要素の平方根の逆数を元の係数行列の両側から施す方法などがある．

スケーリングでは，予め(6)の演算を行い，線形方程式そのものを変形する．従って，求解アルゴリズムに入力する情報（係数行列，右辺項）は，(6)のとおり変形された後の系の情報である．

4. 求解アルゴリズムの体系的な性能比較

本節では，性能情報 DB に格納されている全てのデータに対して，それらの結果を視覚的に表示する体系的な性能比較の方法を説明し，それを用いて新たに確認できる事象について述べる．

4.1 絶対的な指標を用いた収束状況の比較

ここでは，スケーリングのもたらす効果を絶対的な指標で表現するために，反復アルゴリズムが数値解に収束するまでの所要反復回数自体をクラス分けした．

$$\text{Score} = 10 - \left[\frac{\text{所要反復回数} - 1}{\text{係数行列のサイズ}} \times 10 \right] \qquad (7)$$

ここで，[ ]はガウスの記号である．つまり，係数行列のサイズの数値を 10 等分にクラス分けして，所要反復回数が少ないクラスから Score=10, 9, 8, …, 1 と指標を定義する．ここで係数行列のサイズ n を基にした理由は，Krylov 部分空間を張る非定常反復法では，高々 n 回の反復で解に収束することに基づく．より詳細に評価するためには，20 等分，50 等分などして，各々上位 50%，20%分をクラス分けするのも一手段である(図 1).

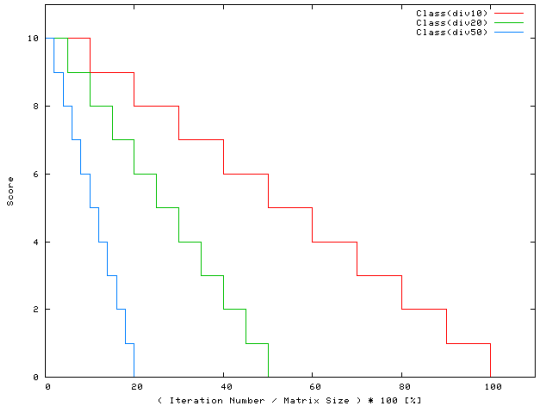

図 1　所要反復回数に対する Score のグラフ

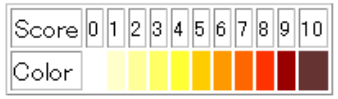

図 2　Score に対する配色

図 1 の 50 等分による性能指標に基づいて，DB 内の求解性能データの性能比較を行った．ここでは，Matrix Market の中から線形方程式向きの下記 52 個の行列を用いた結果を示す．

{1138, 494, 662, 685}_bus, add{20, 30}, arc130, bcsstk{14, 15, 16, 17}, bcsstm26, fs_183_{1, 3, 4, 6}, fs_541_[1-4], fs_680_[1-3], fs_760_[1-3], gr_30_30, gre_1107, gre_216{a, b}, gre__{115, 185, 343, 512}, jpwh_991, lund_{a, b}, memplus, nos[1-7], s1rmq4m1, s1rmt3m1, s2rmq4m1, s2rmt3m1, s3rmq4m1, s3rmt3m1, s3rmt3m3

各々の行列から得られる線形方程式を，Lis の全アルゴリズムを用いて解いたときの所要反復回数に対して，図 1 最左の青色グラフの方針に基づき上位 10 クラスに対して Score を付した．この値に応じ図 2 のようにセル（小枠）を色分けして図示する．

従って，一つの線形方程式に対して収束までの所要反復回数が少なかったアルゴリズムは，濃茶，赤などの濃い色で表現され，所要反復回数が多いほど，黄，薄黄と薄い色で表現されるため，全アルゴリズムの求解性能の傾向を体系的，視覚的に確認できる ．

また，「全く収束していない」場合にはドットを記している．そして，求解アルゴリズム中の残差ベクトルを用いた収束判定(2)では収束しているものの，次式(8)による真の残差ベクトルの相対残差ノルムを評価すると 10^{-8} 以上であった「偽収束」（見かけ上の収束，false convergence）の場合にはアスタリスクを記している．

$$\frac{\|\hat{r}\|_2}{\|b\|_2} = \frac{\|b - A\hat{x}\|_2}{\|b\|_2} \qquad (8)$$

ここで，$\hat{x}$ は求解アルゴリズムにより得られた数値解，$\hat{r}$ は数値解から算出した真の残差であり，係数行列 A と右辺項ベクトル b については，スケーリングや狭義の前処理（以後，本節では前処理とは狭義の方を指す）の適用の有無に関係無く，式(1)で表される係数行列と右辺項ベクトルを指している．

次小節で議論する図 3,4 では，図中のフレーム（太い枠）により，本稿で議論する問題点が確認された 2 種類の解法をグループ分けしており，各々のフレームの中に各前処理の結果が並んでいる．本稿ではあらためて，偽収束をグレーで表示した．

4.2 前処理方式と収束判定

本研究で lis-1.1.2 オリジナル版を用いて体系的な性能評価を行っている中で，解法ごとにグループ分けした際，特定の解法のグループ（各解法のグループには 10 種類の前処理を併用した結果を含んでいる）では，図 3 左のとおり，偽収束となる結果が多く確認された[a]．しかし，これらと同一のアルゴリズムを用いて同じ問題を解く前に，(6)のスケーリングを施した場合の結果が図 3 右である．明らかに右図の方が，偽収束

[a] 本稿で述べる lis-1.1.2 における BiCGStab(*l*)法と TFQMR 法の不具合については，1.2.0 版以後では改善されている．また，本稿内での偽収束の事例はこれら 2 つの解法に顕著な特性ということでは無い．

Vol.2009-HPC-121 No.5
2009/8/4

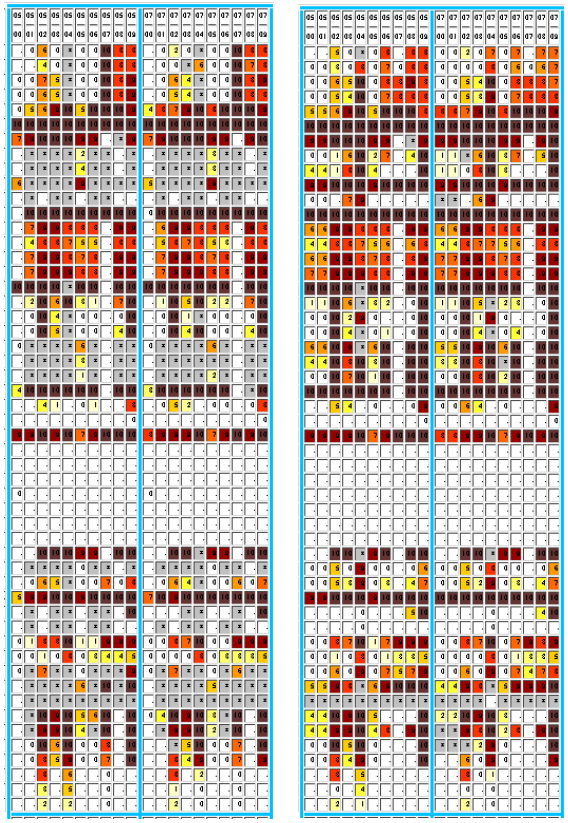

図 3 左：左前処理，右：左前処理とスケーリング併用

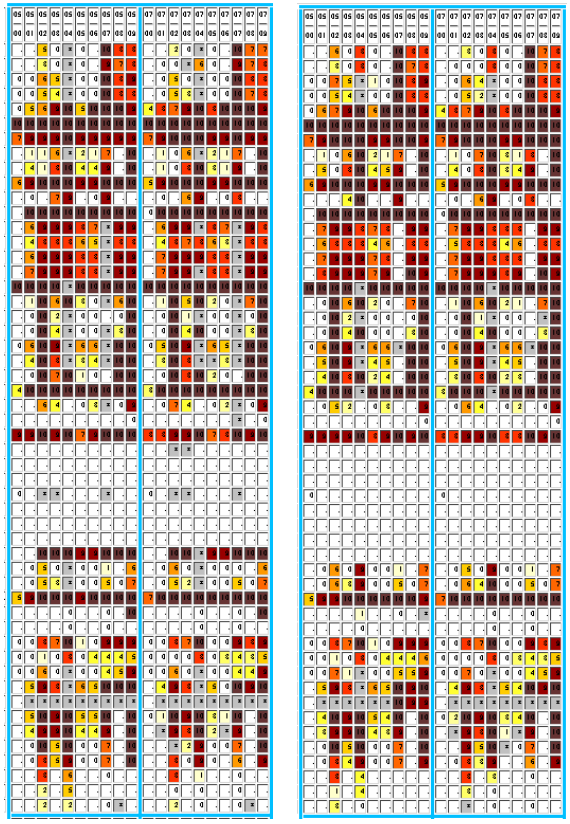

図 4 左：左前処理で収束判定は(12)，右：右前処理

が少ない．一見，スケーリングを併用した効果により収束性が改善されたようにも見えるが，掘り下げて分析した．

求解したときに偽収束が多発した解法(図 3 左)では，左前処理を用いていた．ここで，3 節で述べた前処理方向(3)～(5)に対する残差ベクトルについて確認すると，

$$r_k^{\mathrm{BOTH}} = K_L^{-1}(b - Ax_k) \tag{9}$$

$$r_k^{\mathrm{RIGHT}} = b - Ax_k \tag{10}$$

$$r_k^{\mathrm{LEFT}} = K^{-1}(b - Ax_k), \tag{11}$$

と表わされるが，本来評価すべき残差ベクトルの情報を保持しているのは(10)の右前処理(RIGHT)の系のみであり，(11)の左前処理(LEFT)と(9)の両側前処理(BOTH)とは，

本来評価すべき情報とは異なっている[8].

従って，アルゴリズム中の収束判定式(2)においても，左前処理の場合，分子が(11)に基づく値となる．そこで，前処理後の系に応じた，より良い収束判定式について検討した．まず，左前処理の系に対する収束判定式として，式(12)を用いた結果を図 4 左に示す．さらに，右前処理を用い，(2)による収束判定を行なった結果を図 4 右に示す．

$$\|r_k\|_2 / \|K^{-1}b\|_2 = \|K^{-1}(b - Ax_k)\|_2 / \|K^{-1}b\|_2 \le \varepsilon \qquad (12)$$

図 4 の両図で注目すべきは，左前処理である図 3 左よりも偽収束が減少しており，より安定に求解されているということである．さらに，図 3 左の場合には，アルゴリズム中の収束判定が不完全であったために，偽収束が多発している．

ここで再度，図 3 右のスケーリングを施した上に左前処理アルゴリズムを適用した結果に戻る．そこでの収束判定式とは(13)のとおりである．

$$\|K^{-1}D^{-1}(b - Ax_k)\|_2 / \|D^{-1}b\|_2 \le \varepsilon \qquad (13)$$

これについても，本来の収束判定式(2)とは異なる値で評価していることが分かる．特に，左前処理の系を忠実に表現すると，式(12)に基づくべきであり，スケーリングを併用した時, 分母は $\|K^{-1}D^{-1}b\|_2$ のはずだが, (13)のままでも偽収束が減少している．すなわち，ここでのスケーリングの効果として，収束判定に対する影響という一面も考えられる．

以上をスケーリングの効果と見做すかどうかは状況にも依る（この手段以外では求解不可能であれば仕方が無いのかも知れない）が，前処理系と収束判定の関係という点では，非合理的な問題点を含めたままであることにも留意しておくべきである．

4.3 対称系に対する前処理付き共役勾配法の場合

さて, 前処理付き共役勾配(PCG: Preconditioned Conjugate Gradient)法について考えてみる．PCG 法は対称正定値 (SPD: Symmetric Positive Definite) 行列を係数行列にもつ線形方程式に対する解法だが, 前処理を施した系でも SPD 性を保持するために, 通常, 両側前処理を用いる．しかし，このときには，上述のとおり，アルゴリズム中の残差ベクトルは本来の残差とは異なり，PCG 法では正確な残差を算出できないことになる．

つまり，式(3)において，$K = LL^T$ のコレスキー分解に基づき，収束判定を

$$\|r_k\|_2 / \|L^{-1}b\|_2 = \|L^{-1}(b - Ax_k)\|_2 / \|L^{-1}b\|_2 \le \varepsilon \qquad (14)$$

と変更した場合，PCG 法では，この程度にしか収束判定できないのであろうかという疑問も出てくる．

ここで, CG 法に対する右前処理を考えてみる. 次のように, SPD である行列 K^{-1} を計量(metric)とする内積

$$(u, v)_{K^{-1}} = (u, K^{-1}v), \quad u, v: \text{任意のベクトル} \qquad (15)$$

を定義すると， AK^{-1} はこの内積に関して自己随伴となる[7][8]．すなわち，

$$\begin{aligned}(AK^{-1}u, v)_{K^{-1}} &= (AK^{-1}u, K^{-1}v) = (K^{-1}u, AK^{-1}v) \\ &= (u, K^{-1}AK^{-1}v) = (u, AK^{-1}v)_{K^{-1}}\end{aligned} \qquad (16)$$

が成り立つ[b]．これを用いて，右前処理付き CG 法を導出すると，最終的に，両側前処理付き CG 法と同一のアルゴリズムが導出される[8].

以上から，PCG 法のアルゴリズム中の残差ベクトルに関する内積では式(17)に基づく（r^{RIGHT} は(10)を用いる）が，収束判定では式(18)を用いれば良いので，結局，PCG 法の収束判定では前処理の方向に依存しないことが分かる．

$$\begin{aligned}(r^{\mathrm{BOTH}}, r^{\mathrm{BOTH}})_I &= (r^{\mathrm{RIGHT}}, r^{\mathrm{RIGHT}})_{K^{-1}} \\ &= ((b - Ax), K^{-1}(b - Ax)) \equiv (r, K^{-1}r),\end{aligned} \qquad (17)$$

$$(r, r)_I = (b - Ax, b - Ax) \equiv (r, r). \qquad (18)$$

5. Krylov 部分空間法の２系統の左前処理系

本節では，Krylov 部分空間法の前処理付きアルゴリズムの構築方法に関する，盲点と思われる事柄について指摘する．これは，上述の前処理系と収束判定との不整合の原因の１つとして挙げられる．明確な指摘や言及がなされていない事実の１つであるが，Krylov 部分空間法に対する左前処理系のアルゴリズムは２系統存在することも盲

[b] 同様に，PCG 法の左前処理では，SPD である行列 K を計量とする内積を定義すると，$K^{-1}A$ はその内積に関して自己随伴となり，最終的に両側前処理付き CG 法と同一のアルゴリズムが導出される[8].

点の１つである．ここでは CGS(Conjugate Gradient Squared)法[9]を例に取り説明する．

5.1 両側前処理系に対する前処理付き CGS 法

最初に，前処理無し CGS 法のアルゴリズムと，それに基づいて両側前処理系(3)に基づく前処理付き CGS 法を導出する．本小節でのアルゴリズムの記述や前処理変換については，文献[10]に従った．ただし，α_k と β_k の算出で用いる $\langle u,v\rangle$ の演算は，スカラー積[c]である[11].

$$
\begin{aligned}
&x_0, r_0 = b - Ax_0, r_0^{\#}, \beta_{-1} = 0,\\
&k = 0,1,2,\cdots;\mathrm{Do}\\
&\quad p_k = r_k + \beta_{k-1} z_{k-1},\\
&\quad u_k = p_k + \beta_{k-1}(z_{k-1} + \beta_{k-1} u_{k-1}),\\
&\quad \alpha_k = \left\langle r_0^{\#}, r_k \right\rangle \Big/ \left\langle r_0^{\#}, Au_k \right\rangle,\\
&\quad z_k = p_k - \alpha_k A u_k,\\
&\quad x_{k+1} = x_k + \alpha_k (p_k + z_k),\\
&\quad r_{k+1} = r_k - \alpha_k A(p_k + z_k),\\
&\quad \beta_k = \left\langle r_0^{\#}, r_{k+1} \right\rangle \Big/ \left\langle r_0^{\#}, r_k \right\rangle,\\
&\mathrm{End\ Do}
\end{aligned}
$$

Algorithm 1 前処理無し CGS 法

Alg.1 に対して(19)のように変換すると両側前処理付き CGS 法が得られ，そのときの残差ベクトルは(20)である．ここで⇒記号は，前処理による変換の書換えを表わす．

$$
\begin{aligned}
&\left(K_L^{-1} A K_R^{-1}\right)\left(K_R x\right) = K_L^{-1} b,\\
&\tilde{p} \Rightarrow K_L^{-1} p, \tilde{u} \Rightarrow K_L^{-1} u, \tilde{z} \Rightarrow K_L^{-1} z, \tilde{r} \Rightarrow K_L^{-1} r, \tilde{r}_0^{\#} \Rightarrow K_L^{T} r_0^{\#}
\end{aligned} \tag{19}
$$

$$
\tilde{r} \Rightarrow K_L^{-1} r = K_L^{-1}\left(b - Ax\right) \tag{20}
$$

この変換の途中過程を表わしたものが Alg.2 である．

$$
\begin{aligned}
&x_0, r_0 = b - Ax_0, r_0^{\#}, \beta_{-1} = 0,\\
&k = 0,1,2,\cdots;\mathrm{Do}\\
&\quad K_L^{-1} p_k = K_L^{-1} r_k + \beta_{k-1} K_L^{-1} z_{k-1},\\
&\quad K_L^{-1} u_k = K_L^{-1} p_k + \beta_{k-1} K_L^{-1}(z_{k-1} + \beta_{k-1} u_{k-1}),\\
&\quad \alpha_k = \left\langle K_L^{T} r_0^{\#}, K_L^{-1} r_k \right\rangle \Big/ \left\langle K_L^{T} r_0^{\#}, K_L^{-1} A K_R^{-1} K_L^{-1} u_k \right\rangle,\\
&\quad K_L^{-1} z_k = K_L^{-1} p_k - \alpha_k K_L^{-1} A K_R^{-1} K_L^{-1} u_k,\\
&\quad K_R x_{k+1} = K_R x_k + \alpha_k K_L^{-1}(p_k + z_k),\\
&\quad K_L^{-1} r_{k+1} = K_L^{-1} r_k - \alpha_k K_L^{-1} A K_R^{-1} K_L^{-1}(p_k + z_k),\\
&\quad \beta_k = \left\langle K_L^{T} r_0^{\#}, K_L^{-1} r_{k+1} \right\rangle \Big/ \left\langle K_L^{T} r_0^{\#}, K_L^{-1} r_k \right\rangle,\\
&\mathrm{End\ Do}
\end{aligned}
$$

Algorithm 2 両側前処理系に対する CGS 法（変換途中）

この Alg.2 の式変形を進めると，最終的に Alg.3 が得られる．

$$
\begin{aligned}
&x_0, r_0 = b - Ax_0, r_0^{\#}, \beta_{-1} = 0,\\
&k = 0,1,2,\cdots;\mathrm{Do}\\
&\quad p_k = r_k + \beta_{k-1} z_{k-1},\\
&\quad u_k = p_k + \beta_{k-1}(z_{k-1} + \beta_{k-1} u_{k-1}),\\
&\quad \alpha_k = \left\langle r_0^{\#}, r_k \right\rangle \Big/ \left\langle r_0^{\#}, AK^{-1} u_k \right\rangle,\\
&\quad z_k = p_k - \alpha_k A K^{-1} u_k,\\
&\quad x_{k+1} = x_k + \alpha_k K^{-1}(p_k + z_k),\\
&\quad r_{k+1} = r_k - \alpha_k A K^{-1}(p_k + z_k),\\
&\quad \beta_k = \left\langle r_0^{\#}, r_{k+1} \right\rangle \Big/ \left\langle r_0^{\#}, r_k \right\rangle,\\
&\mathrm{End\ Do}
\end{aligned}
$$

Algorithm 3 両側前処理系に対する前処理付き CGS 法

[c] スカラー積とは，双対空間に属する双対性のあるベクトルどうしの積を指す．演算方法自体は内積と同じであるものの，計量を伴わず，内積の性質は成り立たない．

5.2 両側前処理系に基づく左右前処理付き CGS 法

まず，右前処理系について考える．(19)において $K_L = I, K_R = K$ として Alg.2 を式変形すると，最終的に Alg.3 に辿り着く．ここでの変換式の内，残差ベクトルは，

$$\tilde{r} \Rightarrow r = b - Ax \tag{21}$$

に基づいて構成される．

次に，左前処理系について考える．(19)において $K_L = K, K_R = I$ として Alg.2 を式変形すると，やはり，最終的に Alg.3 に辿り着く．ここでの変換式の残差ベクトルは，

$$\tilde{r} \Rightarrow K^{-1}r = K^{-1}(b - Ax) \tag{22}$$

に基づいて構成され，結局，r 自体は(21)と一致する．

以上から，CGS 法に対する前処理付きアルゴリズムでは，左右両側の前処理方向に関わらず，同一の前処理付きアルゴリズムが得られることが確認された．また，(20)～(22)から，残差ベクトル r 自体も前処理行列の影響を受けていないことが分かる．すなわち，アルゴリズム中の残差ベクトルに対する収束判定式としては，(2)を用いれば良い．

5.3 左前処理系に基づく一般的な左前処理付き CGS 法

ここでは改めて，左前処理系(5)の係数行列と右辺項を Alg.1 に直接適用して導出することにする．すなわち，

$$K^{-1}Ax = K^{-1}b, \tilde{p} \Rightarrow p, \tilde{u} \Rightarrow u, \tilde{z} \Rightarrow z, \tilde{r} \Rightarrow r, \tilde{r}_0^{\#} \Rightarrow r_0^{\#} \tag{23}$$

としたとき，次の Alg.4 が得られる．これが一般に“左前処理付き CGS 法”と呼ばれるものであり，4.2 節で議論した左前処理もこちらである[d].

$x_0, r_0 = K^{-1}(b - Ax_0), r_0^{\#}, \beta_{-1} = 0,$
$k = 0,1,2,\cdots;$ Do
$p_k = r_k + \beta_{k-1} z_{k-1},$
$u_k = p_k + \beta_{k-1}(z_{k-1} + \beta_{k-1} u_{k-1}),$
$\alpha_k = \langle r_0^{\#}, r_k \rangle / \langle r_0^{\#}, K^{-1}Au_k \rangle,$
$z_k = p_k - \alpha_k K^{-1} Au_k,$
$x_{k+1} = x_k + \alpha_k (p_k + z_k),$
$r_{k+1} = r_k - \alpha_k K^{-1} A(p_k + z_k),$
$\beta_k = \langle r_0^{\#}, r_{k+1} \rangle / \langle r_0^{\#}, r_k \rangle,$
End Do

Algorithm 4 左前処理系に基づく一般的な左前処理付き CGS 法

ここで注意すべきことは，残差ベクトルの変換式が，

$$\tilde{r} \Rightarrow r = K^{-1}(b - Ax) \tag{24}$$

に基づいて構成されていることである．これは，(11)の残差ベクトルと同一である．つまり，Alg.4 での収束判定式では，(12)を用いる方が系に対して素直であり，(2)では偽収束を引き起こす可能性が高い．

6. まとめ

本稿では，テスト行列に Matrix Market を用いて線形方程式を用意し，求解ライブラリに Lis-1.1.2 を用いて体系的な性能比較を行い，反復アルゴリズムの収束までの所要反復回数について絶対的な指標により収束状況を可視的に評価する方法を導入した．

体系的性能評価をとおして，特定の解法に多く見られた偽収束に焦点を当てた．一因として，求解アルゴリズム中で用いられる残差ベクトルに対する前処理方式の影響について議論した．そして，前処理方式と収束判定との関係についても体系的な性能評価とともに示した．ここで重要なことは，“前処理方式”と“前処理系の残差ベクトルの定義”と“収束判定”との間の関係，および，“スケーリングが収束判定に及ぼす影響”の議論であり，どの線形方程式において偽収束が起き易いかということは別の

[d] 単に前処理付き CGS 法というと，この Alg.4 を指す場合もある．

Vol.2009-HPC-121 No.5
2009/8/4

議論である．また，このような観点からの問題発見は，体系的性能評価のマトリックス図を描画してようやく見出せるものであり[4]，個々の線形方程式の求解や，個々のアルゴリズムの議論のみでは見出し難い事象である．

さらに，CG 法の場合には計量を定義した内積を用いることで，前処理方向に関わらず同一の前処理付きアルゴリズムが得られることを示した．そして，CGS 法を例に取って２系統の左前処理を示し，各々の前処理系での残差ベクトルの違いを示し，収束判定との関係を説明し，偽収束を引き起こし易い前処理系について述べた．一方，偽収束を引き起こしにくい CGS 法の前処理系では，前処理方向に関わらず同一の前処理付きアルゴリズムが得られることを示した．今後は，他の Krylov 部分空間法についても，アルゴリズムのロジック面も交えて体系的に検証する．

以上の議論は，求解アルゴリズムの設計や数値計算ライブラリ作成における品質管理(QC)や品質保証(QA)の面などで重要である[4].

謝辞　本研究の一部は，文部科学省科研費基盤研究(B)（研究課題番号：20300007）および理化学研究所「次世代生命体統合シミュレーションソフトウェアの研究開発」プロジェクトによるものである．

参考文献

1) Barrett, R., et al., Templates for the solution of linear systems: Building Blocks for Iterative Methods, SIAM, 1994.
2) Itoh, S., Kotakemori, H. and Hasegawa, H., Development of evaluation system for numerical algorithms to solve linear equations, Recent Progress in Scientific Computing (Eds., Wenbin Liu, et al.), Science Press, Beijing, pp.231-241, (2007).
3) 伊藤祥司，体系的評価から見た線形方程式求解に対する前処理の実効性能，情処研報 2008-HPC-116, pp.115-119, (2008).
4) 伊藤祥司，杉原正顯，線形方程式求解プロセスの体系的性能評価に対する品質管理としての考察，日本応用数理学会 2008 年度年会講演予稿集，pp.141-142, (2008).
5) Lis, http://www.ssisc.org/lis/
6) Matrix Market, http://math.nist.gov/MatrixMarket/
7) 森正武，杉原正顯，室田一雄，岩波講座 応用数学 「線形計算」，岩波書店，1994.
8) Saad, Y., Iterative Methods for Sparse Linear Systems, 2nd ed., SIAM, 2003.
9) Sonneveld, P., CGS, A fast Lanczos-type solver for nonsymmetric linear systems, SIAM J. Sci. Statist. Comput., 10, pp. 36–52 (1989).
10) 藤野清次，張紹良，反復法の数理，朝倉書店，1996.
11) 伊理正夫，岩波講座 応用数学「線形代数 II」，岩波書店，1997.