情報処理学会論文誌　コンピューティングシステム　Vol. 3　No. 2　9–19 (June 2010)

クリロフ部分空間法に対する前処理方式と収束判定について

伊藤祥司[†1]　杉原正顯[†2]　姫野龍太郎[†3]

線形方程式求解アルゴリズムの体系的な性能評価により確認された，スケーリングも含めた前処理方式に関わる収束状況の違いの原因について議論する．そして，前処理方向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指摘する．さらに，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複数の系統が存在することを説明し，各々と収束判定との関係について議論する．

Relations between Preconditioning Ways and Convergence Criteria for Krylov Subspace Methods

Shoji Itoh,[†1] Masaaki Sugihara[†2]
and Ryutaro Himeno[†3]

Systematic performance evaluation to compare numerical algorithms of linear equations is introduced and some cause of the difference of convergence status brought by preconditioning system and scaling are discussed. In this paper, especially, the close relations among preconditioning direction, its convergence criterion and definition of residual vector are pointed out. Further, the existence of right or left side preconditioning for the conjugate gradient method despite of symmetric system and the existence of plural types of left preconditioning system for non-symmetric system are explained. Finally, the relations between these events and their convergence criteria are discussed.

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さかんである．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

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{1}$$

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)]．さらに，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きれないことも少なくない．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が行われ，そこで得られた数値計算結果のデータ分析を通して新しい知見が見出されてきている[4),6)]．

また，現在進行中の国家プロジェクトである次世代スーパ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，逐次計算にも共通する，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の場合と比較してはるかに複雑である．したがって，逐次計算の段階ですでに潜在する“求解品質”を大きく揺さぶる問題点の発見と解決が必要である．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重要である[5)]．

本稿では，従来から十分に認識されているとはいい難い，クリロフ部分空間法に前処理を施した系と収束判定との関係について議論する．体系的な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する．特にスケーリングの思わぬ影響を確認したので，それについても議論する．さらに，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，および，一般的にあまり知られていないこととして，非対称系の左前処理アルゴリズムは，残差ベクトルの前処理変換に対応して複数の系統が存在することを説明したうえで，これらの前処理付きアルゴリズムと収束判定と

†1 東京大学情報基盤センター
Information Technology Center, The University of Tokyo
†2 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
†3 理化学研究所情報基盤センター
Advanced Center for Computing and Communication, RIKEN

の関係についても議論する．

2. 線形方程式求解アルゴリズムの前処理

本稿では，線形方程式をクリロフ部分空間法により解く場合に併用する前処理として，次の 2 つについて考える．

■ **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いう場合，こちらを指す．

$K \approx A$ となる前処理行列 K を作り，$K = K_L K_R$ と分解し，

$$(K_L^{-1} A K_R^{-1})(K_R \boldsymbol{x}) = K_L^{-1} \boldsymbol{b} \tag{2}$$

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，式 (2) 自体は実施せず，方程式 (2) 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．したがって，求解アルゴリズムに入力する情報（係数行列，右辺項）は，式 (1) で用いられている情報のままである．

式 (2) 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．

$$K_L = K, K_R = I \tag{3}$$

としたものは左前処理（I は単位行列），

$$K_L = I, K_R = K \tag{4}$$

としたものは右前処理と呼ばれ[1),2),12),15)]，各々は，

$$K^{-1} A \boldsymbol{x} = K^{-1} \boldsymbol{b}, \tag{5}$$

$$(A K^{-1})(K \boldsymbol{x}) = \boldsymbol{b} \tag{6}$$

と式 (1) を変換する．

■ **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 A の対角要素 $D = \mathrm{diag}(A)$ の逆数から構成される行列を施す方法

$$D^{-1} A \boldsymbol{x} = D^{-1} \boldsymbol{b} \tag{7}$$

のほかに，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，求解前に式 (7) の演算を行い，線形方程式そのものを変形する．したがって，求解アルゴリズムに入力する情報（係数行列，右辺項）は，式 (7) のとおり変形された後の系の情報である．

3. 求解アルゴリズムの体系的な性能評価

本章では，性能情報 DB の構築と，DB に格納されているデータに対し，それらの結果を視覚的に表示する体系的な性能評価の方法を説明する．そして，それを用いて新たに確認できる事象について述べる．

3.1 性能情報 DB の構築環境について

数多くの線形方程式を多くの求解アルゴリズムにより計算サーバで解き，そこで生成される様々な求解性能を示すデータを DB に蓄積した[4)]．

線形方程式のテスト問題として，Matrix Market[10)] の中から線形方程式向きの行列を用いて求解問題を用意した．右辺項のベクトルは，解ベクトルの全要素を 1.0 とし式 (1) に代入して生成した．

求解アルゴリズムは，数値計算ライブラリ Lis（Library of Iterative Solvers for Linear Systems)[8)] のバージョン 1.1.2 逐次コードを倍精度で使用し，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．Lis の反復解法 21 種類，前処理 10 種類と 3 種類のスケーリングとのすべての組合せにより計算実行した．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ータ値は，SAAMG のみ “`-saamg_unsym true`” を指定し，それ以外は Lis のデフォルト値である．各アルゴリズムに与える初期解は，$\boldsymbol{x}_0 = \boldsymbol{0}$ とした．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

$$\frac{\|\boldsymbol{r}_k\|_2}{\|\boldsymbol{b}\|_2} \leq 1.0 \times 10^{-12} \ (\equiv \varepsilon) \tag{8}$$

を満たすときとした（$\boldsymbol{r}_k$ 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，典型的な評価方法のうちの 1 つである．計算サーバの環境は，**表 1** に示したとおりである．

3.2 絶対的な指標を用いた収束状況の比較

上述の DB 内の求解性能データを用いて，アルゴリズムの体系的な性能評価を行った．

体系的性能評価を表すマトリックス図[5)]（**図 1**，**図 2**）では，フレーム（太い青枠）により，4 種類の解法をグループ分けしている．また，各々のフレーム内には 10 種類の前処理を併用した結果を，セル（性能指標を表示した小枠）として同じ順序で並べている．本稿で

表 1　計算サーバの環境
Table 1　The environment of computing server.

計算サーバ	HP ProLiant DL585 G2
CPU	AMD Opteron 8220, 2.8 GHz x86_64
メモリサイズ	64 GB
OS	RedHat Linux 2.6.9-42.ELsmp
コンパイラ	Intel icc 10.1, ifort 10.1

表 2　求解アルゴリズム
Table 2　The solution algorithms.

解法	CGS[03], BiCGStab[04], BiCGStab($l=2$)[05], TFQMR[07]
前処理	前処理なし[00], 点 Jacobi[01], ILU(0)[02], SSOR[03], Hybrid[04], I+S[05], SAINV[06], SAAMG[07], Crout ILU[08], ILUT[09]

扱う求解アルゴリズム（解法と前処理との組合せ）は，**表 2** のとおりである[⋆1]．ただし，本章での議論は，3.3.2 項を除き，求解アルゴリズムの特性にはよらない．

反復アルゴリズムの性能指標として，数値解に収束するまでの所要反復回数を次式によりクラス分けした．

$$\mathrm{Score} = 10 - \left[\frac{\text{所要反復回数} - 1}{\text{係数行列のサイズ}} \times 10 \right] \tag{9}$$

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 $\mathrm{Score} = 10, 9, 8, \cdots, 1$ 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，クリロフ部分空間を張る非定常反復法では，たかだか n 回の反復で解に収束することに基づく．

セル内には，式 (9) の性能指標を記載し[⋆2]，“まったく収束していない” 場合にはドットを記している．これは，求解アルゴリズム中の残差ベクトルを用いた収束判定で収束しなかった場合であり，“最大反復回数に到達しても収束しない場合” と，“breakdown で反復が停止した場合” とを一括している．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 (8) では収束しているものの，

⋆1 表 2 の中で示した [ ] 内の 2 桁の数字は Lis 内で定義された解法の ID，および，前処理の ID である．図 1，図 2 では，冒頭の 1 行目に解法の ID，2 行目に前処理の ID が記されている．

⋆2 実際のシステムでは Score 値に応じ配色しているが，本稿の議論には直接は関係しないので，グレーを確認しやすくするために，Score 値に関してはすべて無色とした．

表 3　求解問題
Table 3　The solution problems.

{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, jpwh_991, lund_{a,b}, mcca, mcfe, memplus, nos[1-7], s1rmq4m1, s1rmt3m1, s2rmq4m1, s2rmt3m1, s3rmq4m1, s3rmt3m1, s3rmt3m3, sherman[1-5]

$$\frac{\|\hat{\boldsymbol{r}}\|_2}{\|\boldsymbol{b}\|_2} = \frac{\|\boldsymbol{b} - A\hat{\boldsymbol{x}}\|_2}{\|\boldsymbol{b}\|_2} \tag{10}$$

による真の残差ベクトルの相対残差ノルムを評価したとき 10^{-8}（$\equiv \hat{\varepsilon}$）より大きく，収束と見なし難い場合には，セルをグレーに着色した．ここで，$\hat{\boldsymbol{x}}$ は求解により最終的に得られた数値解，$\hat{\boldsymbol{r}}$ は数値解から算出した真の残差ベクトルであり，係数行列 A と右辺項ベクトル $\boldsymbol{b}$ については，スケーリングや狭義の前処理（以後，本章では前処理とは狭義の方を指す）の適用の有無に関係なく，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．

本稿では，**表 3** に示す行列を用いた結果を示す[⋆3]．ただし，本稿での議論は，3.3.2 項を除き，求解問題自体の特性にはよらない．

3.3　前処理方式に応じた収束判定

本節では，文献 6) からさらにテスト用アルゴリズムを追加し，前処理方式と収束判定の関係について議論する．図 1 左のケースでは，グレーの件数が多く確認された．一方で，これらと同一のアルゴリズムを用いて同じ問題を解く際に，式 (7) のスケーリングを施すとグレーの件数が減少した（図 1 右）．一見，スケーリングを併用した効果として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．

本節で議論する項目の一覧は，最終的に**表 4** 内の A〜D のとおりである．アルゴリズム中で用いた表面的な収束判定は，A，B，D では式 (8) の左辺，C では後述する式 (14) の左辺のとおりであるが，各々が実質的に表している情報とは，表 4 の最右列の “実質的な収束判定の内容” のとおりである．

3.3.1　各前処理系の残差ベクトルと収束判定

図 1 左の求解アルゴリズムでは，左前処理を用いていた．ここで，2 章で述べた前処理方向 (2)，(5)，(6) に対する残差ベクトルについて確認すると，理論的には，

⋆3 ここでの問題名の括弧は UNIX の表記スタイルに倣っている．

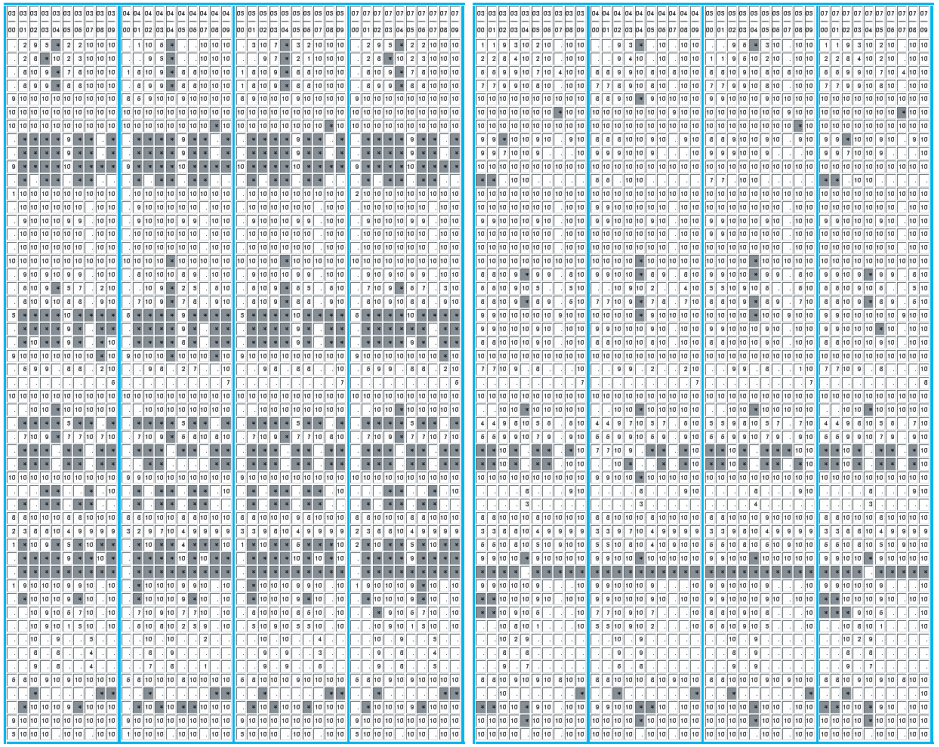

図 1　左：左前処理，右：スケーリングを併用した左前処理
Fig. 1　Left: Left-preconditioning, Right: Left-preconditioning with scaling.

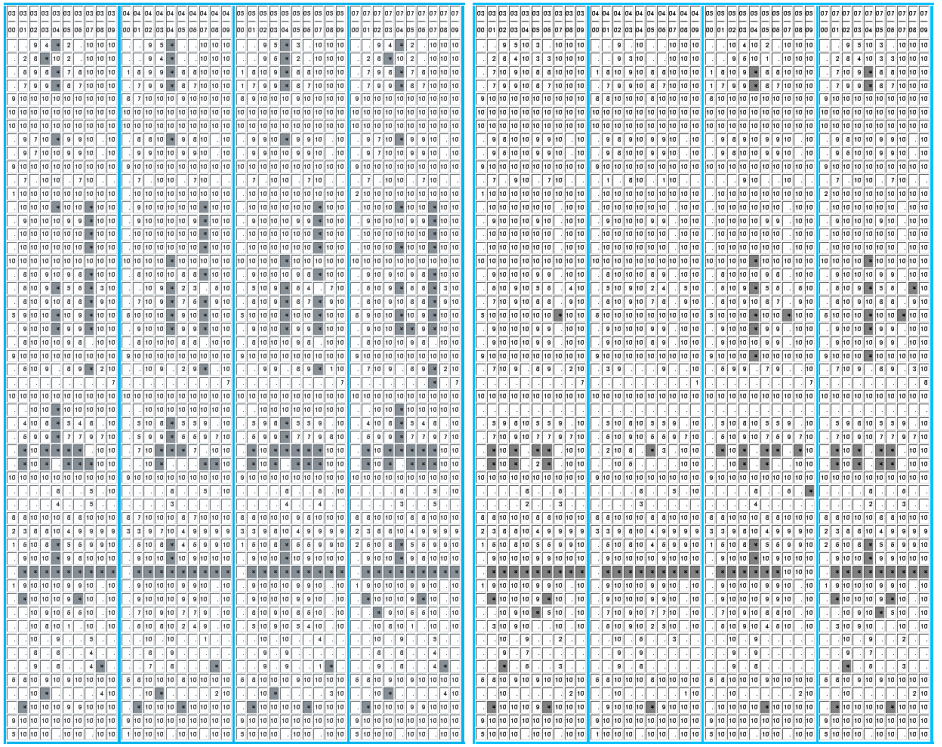

図 2　左：左前処理で収束判定は式 (14)，右：右前処理
Fig. 2　Left: Left-preconditioning judging by (14), Right: Right-preconditioning.

$$\boldsymbol{r}_k^{\mathrm{BOTH}} = K_L^{-1}(\boldsymbol{b} - A\boldsymbol{x}_k), \tag{11}$$
$$\boldsymbol{r}_k^{\mathrm{LEFT}} = K^{-1}(\boldsymbol{b} - A\boldsymbol{x}_k), \tag{12}$$
$$\boldsymbol{r}_k^{\mathrm{RIGHT}} = \boldsymbol{b} - A\boldsymbol{x}_k \tag{13}$$

に相当するが[⋆1]，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式 (13) の右前処理（RIGHT）の系のみであり，式 (11) の両側前処理（BOTH）および式 (12) の左前処理（LEFT）は，本来評価すべき情報とは異なっている[12),15)]．したがって，アルゴリズム中の収束判定式 (8) においても，左前処理の場合，分子が式 (12) に基づく値となってしまっていることが分かる．

そこで，本研究では，前処理後の系に適した収束判定式について検討する．まず，左前処理の系に対しては，右辺項に対して K^{-1} が施されていることから，収束判定式として，

$$\frac{\|\boldsymbol{r}_k\|_2}{\|K^{-1}\boldsymbol{b}\|_2} = \frac{\|K^{-1}(\boldsymbol{b} - A\boldsymbol{x}_k)\|_2}{\|K^{-1}\boldsymbol{b}\|_2} \leq \varepsilon \tag{14}$$

による結果を図 2 左に示す．さらに，右前処理を用い，式 (8) による収束判定を行った結果

⋆1 式 (11)〜(13) の各々の記述は，実際にはアルゴリズム中の残差ベクトル $\boldsymbol{r}_k$ として表現される．本稿の中では，真の残差ベクトル算出の表現については，k 回目の反復における数値解 $\hat{\boldsymbol{x}}_k$ のように，ハット “ˆ” を付した記号を用いて表している．

を図 2 右に示す[⋆1].

図 2 の両図で注目すべきは，左前処理である図 1 左に対して，グレーの件数が減少しており，より安定に求解されているということである．

ここで再度，図 1 右（スケーリングと左前処理アルゴリズムを併用）の結果に戻る．そこでの収束判定式とは，表面上式 (8) ではあるが，式 (7)，(12) より，結局，

$$\frac{\|K^{-1}D^{-1}(\boldsymbol{b}-A\boldsymbol{x}_k)\|_2}{\|D^{-1}\boldsymbol{b}\|_2} \leq \varepsilon \tag{15}$$

である．これについても，本来の収束判定式 (8) 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 (14) に基づくべきであり，スケーリングを併用したとき，分母は $\|K^{-1}D^{-1}\boldsymbol{b}\|_2$ のはずだが，式 (15) のままでもグレーの件数が減少している．すなわち，スケーリングは収束判定に対しても影響を及ぼしたと考えられる．これをスケーリングの有効性と見なすかどうかは状況にもよるだろうが（スケーリング併用以外では求解不可能であれば仕方がないのかもしれない），前処理系と収束判定の関係という点では，非合理的な問題点が潜在していることにも留意するべきである．

以上から，右前処理系の場合のみ，残差ベクトルと収束判定とも “好ましい” 条件である．ここで，“好ましい” とは，前処理変換 K^{-1} やスケーリング D^{-1} の影響を受けず，本来の評価に適した情報を形成している状態を指す．そして，図 1 左とは “甘い収束判定” が原因であったことと，スケーリングが収束判定におよぼす思わぬ影響により “甘い収束判定” が顕著にならない事例もあることが確認された．

3.3.2　各前処理系での収束の振舞い

表 4 の A〜D の各ケースに対し，収束の振舞いはどうであろうか．例題として，体系的評価の問題（表 3）の中から，`nos6` と `fs_680_1` を取り上げ，CGS 法と組み合わせる前処理として点 Jacobi(PJ) 法，ILU(0) 法の 2 つを用いて議論する．

ここでは，Lis 以外に Matlab7.8.0(R2009a) も用いた．本稿で示す図は Matlab による結果であるが，Lis でもほぼ同様の結果を示している[⋆2]．ここでは，Netlib[9] で配布されている SIAM Templates[1] の Matlab コード `cgs.m`（このコードは表 4 のケース D に相当）を基にして各コードを作成した．各々の相対残差ノルムの値が ε 以下になるまで反復を繰り返した．

表 4　各ケースの収束判定
Table 4　Convergence criterion of each case.

	図 1, 図 2 との対応	前処理	アルゴリズム中で用いた収束判定	実質的な収束判定の内容
A	図 1 左	左前処理	$\frac{\|\boldsymbol{r}_k\|_2}{\|\boldsymbol{b}\|_2}$	$\frac{\|K^{-1}(\boldsymbol{b}-A\boldsymbol{x}_k)\|_2}{\|\boldsymbol{b}\|_2}$
B	図 1 右	Scaling ＋左前処理	$\frac{\|\boldsymbol{r}_k\|_2}{\|\check{\boldsymbol{b}}\|_2}$	$\frac{\|K^{-1}D^{-1}(\boldsymbol{b}-A\boldsymbol{x}_k)\|_2}{\|D^{-1}\boldsymbol{b}\|_2}$
C	図 2 左	左前処理	$\frac{\|\boldsymbol{r}_k\|_2}{\|K^{-1}\boldsymbol{b}\|_2}$	$\frac{\|K^{-1}(\boldsymbol{b}-A\boldsymbol{x}_k)\|_2}{\|K^{-1}\boldsymbol{b}\|_2}$
D	図 2 右	右前処理	$\frac{\|\boldsymbol{r}_k\|_2}{\|\boldsymbol{b}\|_2}$	$\frac{\|\boldsymbol{b}-A\boldsymbol{x}_k\|_2}{\|\boldsymbol{b}\|_2}$
E	［参考 1］（真の残差）	右前処理	$\frac{\|\boldsymbol{b}-A\hat{\boldsymbol{x}}_k\|_2}{\|\boldsymbol{b}\|_2}$	$\frac{\|\boldsymbol{b}-A\hat{\boldsymbol{x}}_k\|_2}{\|\boldsymbol{b}\|_2}$

B の第 4 列「アルゴリズム中で用いた収束判定」で記した分母の $\check{\boldsymbol{b}}$ とは，スケーリングされた右辺項 $D^{-1}\boldsymbol{b}$ を意味するが，アルゴリズム中では，そのようには明示されない．本来の右辺項 $\boldsymbol{b}$ と区別するため，この記号で表す．

表 5　各ケースと相対残差ノルムの対応
Table 5　Relative residual norm of each case.

	評価情報	表 4 のケースとの対応	前処理	相対残差ノルム
F	真の残差	A，C	左前処理	$\frac{\|\boldsymbol{b}-A\hat{\boldsymbol{x}}_k\|_2}{\|\boldsymbol{b}\|_2}$
G	Scaling 後真の残差	B	Scaling ＋左前処理	$\frac{\|D^{-1}(\boldsymbol{b}-A\hat{\boldsymbol{x}}_k)\|_2}{\|D^{-1}\boldsymbol{b}\|_2}$
H	真の残差	B	Scaling ＋左前処理	$\frac{\|\boldsymbol{b}-A\hat{\boldsymbol{x}}_k\|_2}{\|\boldsymbol{b}\|_2}$
C	［参考 2］	C	左前処理	$\frac{\|K^{-1}(\boldsymbol{b}-A\boldsymbol{x}_k)\|_2}{\|K^{-1}\boldsymbol{b}\|_2}$
E	真の残差	D，E	右前処理	$\frac{\|\boldsymbol{b}-A\hat{\boldsymbol{x}}_k\|_2}{\|\boldsymbol{b}\|_2}$

図 3 は，表 4 に基づき，アルゴリズム中の残差ベクトルから算出した相対残差ノルムの振舞いである．参考比較として，右前処理の真の残差もプロットした（表 4 の E，［参考 1］）．そして，**図 4** として，表 4 の A〜D のケースに対応する真の残差ベクトルによる相対残差ノルムの振舞いを示す．プロットした値は，**表 5** 最右列の「相対残差ノルム」自体である．参考比較として，左前処理のアルゴリズム中の残差ベクトルによる値もプロットした（表 5 の C，［参考 2］）．図 3，図 4 のすべてについて，横軸は反復回数，縦軸は `log` スケールで表された相対残差ノルムの値である．各図上部のタイトルは「問題名：求解アルゴリズム」

⋆1 ここで，判定式の分母を $\|\boldsymbol{r}_0\|_2$ と表すことには問題がある[1]．収束判定が初期解 $\boldsymbol{x}_0$ に依存する，あるいは，初期解は必ず $\boldsymbol{x}_0 = \boldsymbol{0}$ であるという前提条件が新たに必要となるためである．

⋆2 ILU(0) 前処理では，Lis と Matlab とのプログラミングレベルでの詳細設計の違いによると思われる収束状況の多少の相違はあるが，収束の傾向はおおむね同じである．

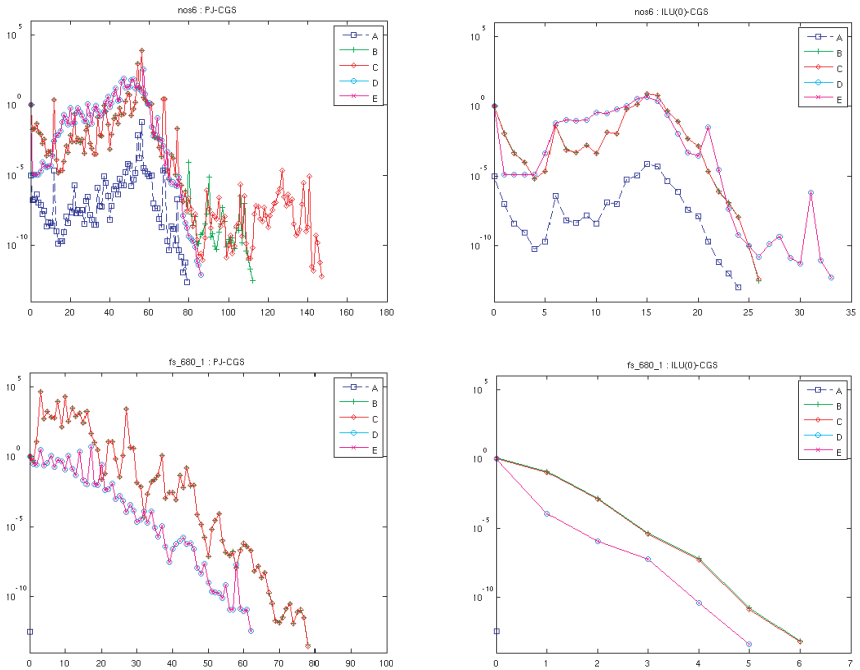

図 3　アルゴリズムの相対残差ノルムの振舞い
Fig. 3　Behavior of algorithm's relative residual norm.

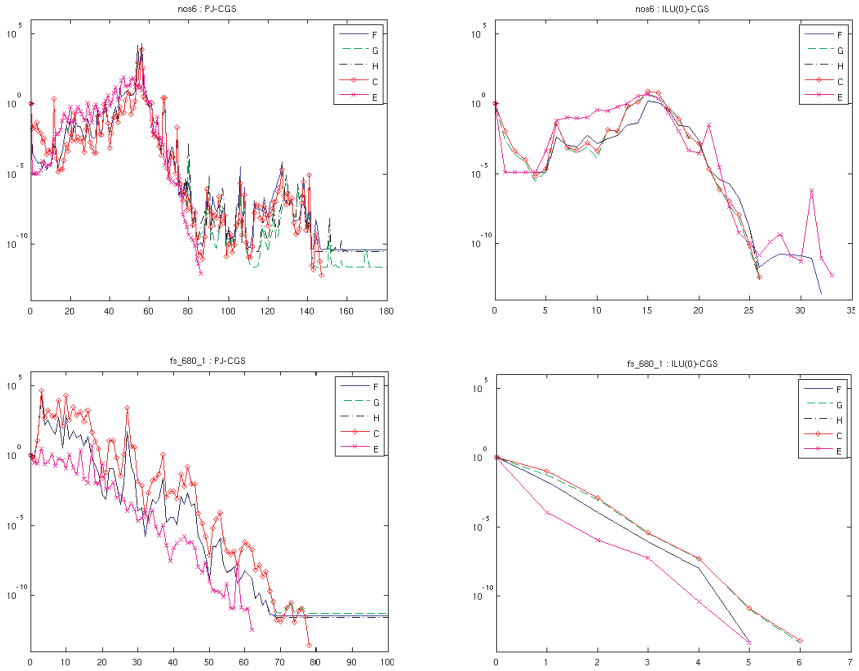

図 4　真の相対残差ノルムの振舞い
Fig. 4　Behavior of true relative residual norm.

である．表 4，表 5 内の記号 $\hat{\boldsymbol{x}}_k$ と $\boldsymbol{b} - A\boldsymbol{x}_k$ の意味は，式 (11)〜(13) に対する注釈と同じである．

図 3 では，A のみ他の収束グラフからまったく離れており，一見，早く収束しているようにも見える．極端なのは，`fs_680_1` の結果（左下，右下の図）では 0 回で収束（縦軸と重なっている）しており，これは初期残差ですでに収束条件を満足してしまっていることを示す[⋆1]．B と C の収束の振舞いはおおむね同じであるが，一般には，用いる前処理や求解問題によって振舞いが異なる[⋆2]．D と E の振舞いはほぼ一致している．

図 4 では，図 3 において振舞いがおおむね同じであった B と C に対応するケースに注目する．まず，F と H の振舞いがほぼ一致しているが，一般には，用いる前処理や求解問題によっては一致しない．左上，左下の図では，F と H は，ともに $\hat{\varepsilon}$ より下の値で停滞している．右上の図で収束点が異なるのは，スケーリングのわずかな影響である．次に，G と C（C は B とも振舞いがおおむね同じであり収束しているので，ここでは B の代用として C と比較する）は途中まで振舞いが似ているものの，左上，左下の図では，G は途中から停滞してしまう．つまり，G に対応する B での収束とは，厳密には “偽収束”（見かけ上の収束，false convergence）であるが，$\varepsilon = 10^{-12} \sim \hat{\varepsilon} = 10^{-8}$ の範囲内のため，ここでは収束と判断している．本稿で “偽収束” とは，求解アルゴリズム中の残差ベクトルによる収束

⋆1 Lis では 1 回で収束しているが，これは初期残差での収束判定を行っていないだけである．

⋆2 図 3 の左上のみ，収束の途中から振舞いが異なっている．B では $K = I$，C では $K = D$ と結果的に数学的に同値となるはずだが，B ではあらかじめ $D^{-1}A$ を計算，C では反復中に $\boldsymbol{u} = A\boldsymbol{v}$，$K^{-1}\boldsymbol{u}$ を計算と，演算形態が異なる．

判定（図 1 左に対し議論した “甘い収束判定” によるものを除く）では収束しているが，真の残差ベクトルでは収束せずに停滞する事象を指す．また，左前処理の真の残差をプロットした F（途中から停滞）と比較すると，C の収束も偽収束であることが確認できる．ここでの例題の中で停滞せずに求解できたのは，右前処理の E（すなわち，表 4 の D）のみであった．

本節での議論からも，図 1 左（表 4 の A）では “甘い収束判定” による結果であったことが分かる．これ以外は，あくまで参考データであり，D の右前処理が最良と結論するには及ばないものの，ここでの実験では良好な結果であった．

4. 対称系に対する前処理付き共役勾配法の場合

本章では，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）行列を係数行列に持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は，通常，両側前処理を用いる．しかし，ここで，3.3 節での前処理方向と収束判定の関係の議論に従うと，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 (2) において $K_L = L$，$K_R = L^T$ とした，

$$K = LL^T \tag{16}$$

のコレスキー分解に基づき，収束判定を

$$\frac{\|\boldsymbol{r}_k\|_2}{\|L^{-1}\boldsymbol{b}\|_2} = \frac{\|L^{-1}(\boldsymbol{b} - A\boldsymbol{x}_k)\|_2}{\|L^{-1}\boldsymbol{b}\|_2} \leq \varepsilon \tag{17}$$

と変更し，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．そうなると，CG 法に対しても右前処理を適用したいが，一般には，対称行列と対称行列をかけても対称行列とはならない．そこで，次のように，SPD 行列 K^{-1} を計量（metric）とする内積

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

を定義する．このとき，AK^{-1} はこの内積に関して自己随伴となる[11),12)]．すなわち，

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

であり，AK^{-1} についても対称性に相当する性質が成り立つ．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[12),15),⋆1]．

以下に，前処理なし CG 法のアルゴリズムを著し，その両側前処理 (2) に基づく前処理付き CG 法の導出過程を示したうえで，収束判定について議論する．本章でのアルゴリズムの記述や前処理変換については文献 2) に従った．また，本章の Algorithm 1〜3 で用いられている $\boldsymbol{x}_0$ に対しては，適当な値を与えるものとする．

Algorithm 1. **前処理なし** CG **法：**

$\boldsymbol{x}_0$, $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$, $\beta_{-1} = 0$,

$k = 0, 1, 2, \cdots$, until $\|\boldsymbol{r}_k\|_2 / \|\boldsymbol{b}_k\|_2 \leq \varepsilon$ Do:

$$\begin{aligned}&\boldsymbol{p}_k = \boldsymbol{r}_k + \beta_{k-1}\boldsymbol{p}_{k-1}, \\ &\alpha_k = \frac{(\boldsymbol{r}_k, \boldsymbol{r}_k)}{(\boldsymbol{p}_k, A\boldsymbol{p}_k)}, \\ &\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k\boldsymbol{p}_k, \\ &\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k A\boldsymbol{p}_k, \\ &\beta_k = \frac{(\boldsymbol{r}_{k+1}, \boldsymbol{r}_{k+1})}{(\boldsymbol{r}_k, \boldsymbol{r}_k)},\end{aligned}$$

End Do

Algorithm 1 の初期設定部分以外に対し，

$$\begin{aligned}&(K_L^{-1}AK_R^{-1})(K_R\boldsymbol{x}) = K_L^{-1}\boldsymbol{b}, \\ &\tilde{\boldsymbol{p}} \Rightarrow K_R\boldsymbol{p},\ \ \tilde{\boldsymbol{r}} \Rightarrow K_L^{-1}\boldsymbol{r}\end{aligned} \tag{18}$$

と変換する．ここで $\Rightarrow$ は，前処理による変換の書き換えを表す．そして，内積に対する計量 G を用いて表すと前処理付き CG 法が得られる．この変換の途中状況が Algorithm 2 である．

Algorithm 2. **前処理付き** CG **法（変換途中）：**

$\boldsymbol{x}_0$, $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$, $\beta_{-1} = 0$,

$k = 0, 1, 2, \cdots$, until **収束判定** $1 \leq \varepsilon$ Do:

$$\begin{aligned}&K_R\boldsymbol{p}_k = K_L^{-1}\boldsymbol{r}_k + \beta_{k-1}K_R\boldsymbol{p}_{k-1}, \\ &\alpha_k = \frac{(K_L^{-1}\boldsymbol{r}_k, K_L^{-1}\boldsymbol{r}_k)_G}{(K_R\boldsymbol{p}_k, K_L^{-1}AK_R^{-1}K_R\boldsymbol{p}_k)_G},\end{aligned}$$

⋆1 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．

$$K_R\boldsymbol{x}_{k+1} = K_R\boldsymbol{x}_k + \alpha_k K_R\boldsymbol{p}_k,$$
$$K_L^{-1}\boldsymbol{r}_{k+1} = K_L^{-1}\boldsymbol{r}_k - \alpha_k K_L^{-1}AK_R^{-1}K_R\boldsymbol{p}_k,$$
$$\beta_k = \frac{(K_L^{-1}\boldsymbol{r}_{k+1}, K_L^{-1}\boldsymbol{r}_{k+1})_G}{(K_L^{-1}\boldsymbol{r}_k, K_L^{-1}\boldsymbol{r}_k)_G},$$

End Do

ここで，前処理変換の途中である Alg.2 の “**収束判定** 1” とは，

$$\sqrt{(K_L^{-1}\boldsymbol{r}_k, K_L^{-1}\boldsymbol{r}_k)_G} \Big/ \sqrt{(K_L^{-1}\boldsymbol{b}, K_L^{-1}\boldsymbol{b})_G} \tag{19}$$

である．ここでは，この分子の内積演算に注目して議論する．分母に関しても分子と同様の議論である（$\because \tilde{\boldsymbol{r}} \Rightarrow K_L^{-1}\boldsymbol{r}$ と $\tilde{\boldsymbol{b}} \Rightarrow K_L^{-1}\boldsymbol{b}$ の前処理変換は同じである）．

あらためて，右前処理を考えると，式 (4) の $K_L = I$ と $G = K^{-1}$ を用い，

$$(K_L^{-1}\boldsymbol{r}_k, K_L^{-1}\boldsymbol{r}_k)_G = (\boldsymbol{r}_k, \boldsymbol{r}_k)_{K^{-1}} = (\boldsymbol{r}_k, K^{-1}\boldsymbol{r}_k) \tag{20}$$

である．ところが，これらの内積で算出される残差ベクトルのノルムとは，

$$(\boldsymbol{r}_k, K^{-1}\boldsymbol{r}_k) = (L^{-1}\boldsymbol{r}_k, L^{-1}\boldsymbol{r}_k) = \|L^{-1}\boldsymbol{r}_k\|_2^2 \tag{21}$$

に基づくため，本来評価したい

$$\boldsymbol{r}_k = \boldsymbol{b} - A\boldsymbol{x}_k \tag{22}$$

から算出されるノルムとは異なる値である．

ここで，式 (20) で表した $\boldsymbol{r}_k$ 自体は，右前処理に基づく残差ベクトル (22) である．したがって，収束判定の内積の計量として単位行列を用いると，

$$(\boldsymbol{r}_k, \boldsymbol{r}_k)_I \equiv \|\boldsymbol{r}_k\|_2^2 \tag{23}$$

となり，本来評価したい “好ましい” 条件での収束判定のノルムが得られる．

一方，両側前処理では，式 (16) での $K_L = L$ と $G = I$ を用い，

$$(K_L^{-1}\boldsymbol{r}_k, K_L^{-1}\boldsymbol{r}_k)_G = (L^{-1}\boldsymbol{r}_k, L^{-1}\boldsymbol{r}_k)_I = (\boldsymbol{r}_k, K^{-1}\boldsymbol{r}_k) \tag{24}$$

である．つまり，式 (20) と比較して，CG 法に対する両側前処理と右前処理とでは，各々の計量を定義することにより，残差ベクトルの内積が同一となることが確認された．

ここで，CG 法の両側前処理の場合には，

$$\tilde{r}_k \Rightarrow L^{-1}\boldsymbol{r}_k = L^{-1}(\boldsymbol{b} - A\boldsymbol{x}_k) \tag{25}$$

であり[2)]，すなわち，式 (22) 同様 $\boldsymbol{r}_k = \boldsymbol{b} - A\boldsymbol{x}_k$ と定義したことと同じである．したがって，両側前処理の場合でも，残差ベクトルそのものは前処理行列の影響を受けていないことが分かる．同様に，収束判定に現れる右辺項 $\boldsymbol{b}$ についても前処理行列の影響を受けていない．

結局，PCG 法の収束判定では前処理の方向に影響されず，式 (8) を用いた “好ましい” 収束判定となることが確認された．以上の収束判定の議論もふまえ Algorithm 2 の式変形を進めると，最終的に Algorithm 3 が得られる．

Algorithm 3. **前処理付き** CG **法：**

$\boldsymbol{x}_0$, $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$, $\beta_{-1} = 0$,

$k = 0, 1, 2, \cdots$, until $\|\boldsymbol{r}_k\|_2 / \|\boldsymbol{b}_k\|_2 \le \varepsilon$ Do:

$$\boldsymbol{p}_k = K^{-1}\boldsymbol{r}_k + \beta_{k-1}\boldsymbol{p}_{k-1},$$
$$\alpha_k = \frac{(\boldsymbol{r}_k, K^{-1}\boldsymbol{r}_k)}{(\boldsymbol{p}_k, A\boldsymbol{p}_k)},$$
$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k\boldsymbol{p}_k,$$
$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k A\boldsymbol{p}_k,$$
$$\beta_k = \frac{(\boldsymbol{r}_{k+1}, K^{-1}\boldsymbol{r}_{k+1})}{(\boldsymbol{r}_k, K^{-1}\boldsymbol{r}_k)},$$

End Do

収束判定に関する議論を補足すると，前処理なし CG 法（Algorithm 1）では，アルゴリズム中の α_k や β_k に現れる残差ベクトルの内積

$$(\boldsymbol{r}_k, \boldsymbol{r}_k) \tag{26}$$

の情報をそのまま収束判定に流用できる．しかし，前処理付き CG 法（Algorithm 3）の収束判定では，あらためて式 (26) の内積に基づくノルムを算出する必要がある．

5. クリロフ部分空間法の複数系統の左前処理

本章では，クリロフ部分空間法の前処理付きアルゴリズムの設計に関する，盲点と思われる事柄について指摘する．これは，3.3 節で議論した前処理系と収束判定との不整合を引き起こしかねない原因の 1 つとも思われる．国内外の既出の文献などでも明確な指摘や言及を見かけない事実の 1 つとして，クリロフ部分空間法に対する左前処理のアルゴリズムは複数の系統が存在する．ここでは CGS（Conjugate Gradient Squared）法[13)] を例にとり説明する．

本章での前処理付き CGS 法のアルゴリズムの記述は，オリジナルの文献 13) に記されたものではなく，現在，一般的に認知されている文献 1)，2)，14) の系統に従い，それを最初

に表す[⋆1]．次いで，それとは別系統の左前処理付きアルゴリズムを表す．

5.1　両側前処理付き CGS 法

最初に，前処理なし CGS 法のアルゴリズムを表し，その両側前処理 (2) に基づく前処理付き CGS 法を導出する．本節でのアルゴリズムの記述や前処理変換についても，記述の簡明な文献 2) に従った．ただし，α_k と β_k の算出に用いる $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$ の演算は，スカラー積[⋆2]である[3)]．また，本節の Algorithm 4〜7 で用いられている $\boldsymbol{x}_0$ と $\boldsymbol{r}_0^\sharp$ に対しては，適当な値を与えるものとする．

Algorithm 4. **前処理なし** CGS **法：**

$\boldsymbol{x}_0$, $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$, $\boldsymbol{r}_0^\sharp$, $\beta_{-1} = 0$,

$k = 0, 1, 2, \cdots$, until $\|\boldsymbol{r}_k\|_2 / \|\boldsymbol{b}\|_2 \leq \varepsilon$ Do:

$$\begin{aligned}
&\boldsymbol{p}_k = \boldsymbol{r}_k + \beta_{k-1}\boldsymbol{z}_{k-1},\\
&\boldsymbol{u}_k = \boldsymbol{p}_k + \beta_{k-1}(\boldsymbol{z}_{k-1} + \beta_{k-1}\boldsymbol{u}_{k-1}),\\
&\alpha_k = \frac{\langle \boldsymbol{r}_0^\sharp, \boldsymbol{r}_k \rangle}{\langle \boldsymbol{r}_0^\sharp, A\boldsymbol{u}_k \rangle},\\
&\boldsymbol{z}_k = \boldsymbol{p}_k - \alpha_k A\boldsymbol{u}_k,\\
&\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k(\boldsymbol{p}_k + \boldsymbol{z}_k),\\
&\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k A(\boldsymbol{p}_k + \boldsymbol{z}_k),\\
&\beta_k = \frac{\langle \boldsymbol{r}_0^\sharp, \boldsymbol{r}_{k+1} \rangle}{\langle \boldsymbol{r}_0^\sharp, \boldsymbol{r}_k \rangle},
\end{aligned}$$

End Do

Algorithm 4 の初期設定部分以外に対し，

$$(K_L^{-1}AK_R^{-1})(K_R\boldsymbol{x}) = K_L^{-1}\boldsymbol{b},$$

$$\tilde{\boldsymbol{p}} \Rightarrow K_L^{-1}\boldsymbol{p},\ \tilde{\boldsymbol{u}} \Rightarrow K_L^{-1}\boldsymbol{u},\ \tilde{\boldsymbol{z}} \Rightarrow K_L^{-1}\boldsymbol{z}, \tilde{\boldsymbol{r}} \Rightarrow K_L^{-1}\boldsymbol{r},\ \tilde{\boldsymbol{r}}_0^\sharp \Rightarrow K_L^T\boldsymbol{r}_0^\sharp \tag{27}$$

と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，前処理による変換の書き換え $\Rightarrow$ を用いて，

$$\tilde{\boldsymbol{r}} \Rightarrow K_L^{-1}\boldsymbol{r} = K_L^{-1}(\boldsymbol{b} - A\boldsymbol{x}). \tag{28}$$

したがって，アルゴリズム中の残差ベクトルは $\boldsymbol{r}_k = \boldsymbol{b} - A\boldsymbol{x}_k$ である．この変換の途中状況が Algorithm 5 である．

Algorithm 5. **両側前処理付き** CGS **法（変換途中）：**

$\boldsymbol{x}_0$, $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$, $\boldsymbol{r}_0^\sharp$, $\beta_{-1} = 0$,

$k = 0, 1, 2, \cdots$, until **収束判定** $2 \leq \varepsilon$ Do:

$$\begin{aligned}
&K_L^{-1}\boldsymbol{p}_k = K_L^{-1}\boldsymbol{r}_k + \beta_{k-1}K_L^{-1}\boldsymbol{z}_{k-1},\\
&K_L^{-1}\boldsymbol{u}_k = K_L^{-1}\boldsymbol{p}_k + \beta_{k-1}K_L^{-1}(\boldsymbol{z}_{k-1} + \beta_{k-1}\boldsymbol{u}_{k-1}),\\
&\alpha_k = \frac{\langle K_L^T\boldsymbol{r}_0^\sharp, K_L^{-1}\boldsymbol{r}_k \rangle}{\langle K_L^T\boldsymbol{r}_0^\sharp, K_L^{-1}AK_R^{-1}K_L^{-1}\boldsymbol{u}_k \rangle},\\
&K_L^{-1}\boldsymbol{z}_k = K_L^{-1}\boldsymbol{p}_k - \alpha_k K_L^{-1}AK_R^{-1}K_L^{-1}\boldsymbol{u}_k,\\
&K_R\boldsymbol{x}_{k+1} = K_R\boldsymbol{x}_k + \alpha_k K_L^{-1}(\boldsymbol{p}_k + \boldsymbol{z}_k),\\
&K_L^{-1}\boldsymbol{r}_{k+1} = K_L^{-1}\boldsymbol{r}_k - \alpha_k K_L^{-1}AK_R^{-1}K_L^{-1}(\boldsymbol{p}_k + \boldsymbol{z}_k),\\
&\beta_k = \frac{\langle K_L^T\boldsymbol{r}_0^\sharp, K_L^{-1}\boldsymbol{r}_{k+1} \rangle}{\langle K_L^T\boldsymbol{r}_0^\sharp, K_L^{-1}\boldsymbol{r}_k \rangle},
\end{aligned}$$

End Do

ここで，“**収束判定** 2” についても CG 法の式 (25) での議論に倣うと，式 (28) に対しても同様に結論付けられ，

$$\|\boldsymbol{r}_k\|_2 / \|\boldsymbol{b}\|_2 \tag{29}$$

である．

この式変形を進めると，最終的に Algorithm 6 が得られる．

Algorithm 6. **両側前処理付き** CGS **法：**

$\boldsymbol{x}_0$, $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$, $\boldsymbol{r}_0^\sharp$, $\beta_{-1} = 0$,

$k = 0, 1, 2, \cdots$, until $\|\boldsymbol{r}_k\|_2 / \|\boldsymbol{b}\|_2 \leq \varepsilon$ Do:

$$\begin{aligned}
&\boldsymbol{p}_k = \boldsymbol{r}_k + \beta_{k-1}\boldsymbol{z}_{k-1},\\
&\boldsymbol{u}_k = \boldsymbol{p}_k + \beta_{k-1}(\boldsymbol{z}_{k-1} + \beta_{k-1}\boldsymbol{u}_{k-1}),\\
&\alpha_k = \frac{\langle \boldsymbol{r}_0^\sharp, \boldsymbol{r}_k \rangle}{\langle \boldsymbol{r}_0^\sharp, AK^{-1}\boldsymbol{u}_k \rangle},\\
&\boldsymbol{z}_k = \boldsymbol{p}_k - \alpha_k AK^{-1}\boldsymbol{u}_k,\\
&\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k K^{-1}(\boldsymbol{p}_k + \boldsymbol{z}_k),\\
&\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k AK^{-1}(\boldsymbol{p}_k + \boldsymbol{z}_k),\\
&\beta_k = \frac{\langle \boldsymbol{r}_0^\sharp, \boldsymbol{r}_{k+1} \rangle}{\langle \boldsymbol{r}_0^\sharp, \boldsymbol{r}_k \rangle},
\end{aligned}$$

End Do

⋆1 特に，文献 14) では，CGS の左・右・両側の前処理付きアルゴリズムが一致することに言及している．

⋆2 スカラー積とは，演算方法自体は内積と同じであるものの，計量をともなわず，内積の性質は成り立たない．本稿では特に，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．

5.2　左・右前処理付き CGS 法

本稿での左・右前処理の定義は式 (3)，(4) に従う．まず，右前処理について考える．式 (27) において式 (4) として Algorithm 5 を式変形すると，最終的に Algorithm 6 と同じ記述となった．ここでの変換式の残差ベクトルは，

$$\tilde{\boldsymbol{r}} \Rightarrow \boldsymbol{r} = \boldsymbol{b} - A\boldsymbol{x} \tag{30}$$

に基づいて構成される．次に，左前処理について考える．式 (27) において式 (3) として Algorithm 5 を式変形すると，やはり，最終的に Algorithm 6 と同じ記述になった．ここでの変換式のうち，残差ベクトルは，

$$\tilde{\boldsymbol{r}} \Rightarrow K^{-1}\boldsymbol{r} = K^{-1}(\boldsymbol{b} - A\boldsymbol{x}) \tag{31}$$

に基づいて構成され，結局，$\boldsymbol{r}$ 自体は式 (30) 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かかわらず，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，式 (28)，(30)，(31) から，残差ベクトル $\boldsymbol{r}$ そのものは，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，Algorithm 6 での収束判定式としては式 (8) を用い，“好ましい” 収束判定である．

5.3　一般的な左前処理付き CGS 法

本節では改めて，左前処理 (5) の係数行列と右辺項のみを Algorithm 4 に直接適用して導出してみる．すなわち，

$$K^{-1}A\boldsymbol{x} = K^{-1}\boldsymbol{b},$$
$$\tilde{\boldsymbol{p}} \Rightarrow \boldsymbol{p}, \tilde{\boldsymbol{u}} \Rightarrow \boldsymbol{u}, \tilde{\boldsymbol{z}} \Rightarrow \boldsymbol{z}, \tilde{\boldsymbol{r}} \Rightarrow \boldsymbol{r}, \tilde{\boldsymbol{r}}_0^\sharp \Rightarrow \boldsymbol{r}_0^\sharp \tag{32}$$

としたとき，次の Algorithm 7 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，3.3 節で議論した左前処理とはこちらの方である[⋆1]．

Algorithm 7. **一般的な左前処理付き** CGS **法：**

$\boldsymbol{x}_0$, $\boldsymbol{r}_0 = K^{-1}(\boldsymbol{b} - A\boldsymbol{x}_0)$, $\boldsymbol{r}_0^\sharp$, $\beta_{-1} = 0$,

$k = 0, 1, 2, \cdots$, until $\|\boldsymbol{r}_k\|_2 / \|K^{-1}\boldsymbol{b}\|_2 \leq \varepsilon$ Do:

$$\boldsymbol{p}_k = \boldsymbol{r}_k + \beta_{k-1}\boldsymbol{z}_{k-1},$$
$$\boldsymbol{u}_k = \boldsymbol{p}_k + \beta_{k-1}(\boldsymbol{z}_{k-1} + \beta_{k-1}\boldsymbol{u}_{k-1}),$$
$$\alpha_k = \frac{\langle \boldsymbol{r}_0^\sharp, \boldsymbol{r}_k \rangle}{\langle \boldsymbol{r}_0^\sharp, K^{-1}A\boldsymbol{u}_k \rangle},$$
$$\boldsymbol{z}_k = \boldsymbol{p}_k - \alpha_k K^{-1}A\boldsymbol{u}_k,$$
$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k(\boldsymbol{p}_k + \boldsymbol{z}_k),$$
$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k K^{-1}A(\boldsymbol{p}_k + \boldsymbol{z}_k),$$
$$\beta_k = \frac{\langle \boldsymbol{r}_0^\sharp, \boldsymbol{r}_{k+1} \rangle}{\langle \boldsymbol{r}_0^\sharp, \boldsymbol{r}_k \rangle},$$

End Do

ここで注目すべきことは，残差ベクトルの変換式が，

$$\tilde{\boldsymbol{r}} \Rightarrow \boldsymbol{r} = K^{-1}(\boldsymbol{b} - A\boldsymbol{x}) \tag{33}$$

に基づいて構成されることである．これは，式 (12) の残差ベクトルと同一である．つまり，Algorithm 7 での収束判定式では，式 (14) を用いる方が系に対して素直であり，式 (8) は不適切である．

6. ま と め

体系的な性能評価を行い，反復アルゴリズムの収束までの所要反復回数に基づく指標により収束状況を全体的に評価する方法について説明した．

本稿では，体系的性能評価を通して “スケーリングを含めた前処理方式” と “収束判定” と “各前処理系に応じた残差ベクトルの定義” との関係，および，“スケーリングが収束判定に及ぼす影響” について議論した．ここで重要なことは，これらに着目した前処理付きアルゴリズムに対する数理面からの一般的な議論であり，どの線形方程式において，“甘い判定での収束” や “偽収束” がどの程度起きるのかということは別の議論である．また，このような観点からの問題発見や分析は，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[5]，個々の線形方程式の求解や，個々の求解アルゴリズムの議論のみでは見出し難い事象である．

前処理方向と収束判定の関係について，CG 法の場合には前処理行列に基づく計量をともなう内積を定義することで，前処理方向にかかわらず同一の前処理付きアルゴリズムが得られる[12),15)] ことを説明したうえで，アルゴリズム中の内積とは独立して収束判定の内積を定義することで，収束判定に影響が及ばないことを示した．さらに，CGS 法を例にとり，左前処理の複数の系統について，それらの導出過程も表したうえで，各々の前処理系での残差ベクトルの違いを示した．そして，収束判定との関係を説明し，“甘い収束判定” を引き起こしやすい前処理系について述べた．ここで，“好ましい” 収束判定となる CGS 法の前処理系では，前処理方向にかかわらず同一の前処理付きアルゴリズムが得られる[14)] ことを説

⋆1 単に前処理付き CGS 法というと，Algorithm 7 を指す場合もある．

明したが，すべてのクリロフ部分空間法がこのとおりというわけではない．文献 7) では，他の解法についても考察がなされ，理論体系に照らし合わせた議論を経て新しい知見が得られている．

本稿での議論は，求解アルゴリズムの設計や数値計算プログラム作成における品質管理（QC）や品質保証（QA）の面などでも重要である[5)]．

謝辞　第 1 著者は，Lis の利用にあたり多くの助言をいただいた株式会社サムスン横浜研究所の小武守恒博士に感謝いたします．本研究の一部は，文部科学省科学研究費補助金基盤研究（B）（課題番号：20300007，21300007，21300017）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). 長谷川里美，長谷川秀彦，藤野清次（訳）：反復法 Templates，朝倉書店 (1996).
2) 藤野清次，張　紹良：反復法の数理，朝倉書店 (1996).
3) 伊理正夫：岩波講座 応用数学「線形代数 II」，岩波書店 (1997).
4) Itoh, S., Kotakemori, H. and Hasegawa, H.: Development of evaluation system for numerical algorithms to solve linear equations, *Recent Progress in Scientific Computing*, Liu, W., et al. (Eds.), pp.231–241, Science Press, Beijing (2007).
5) 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).
6) 伊藤祥司，杉原正顯：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，情報処理学会研究報告，Vol.2009-HPC-121, No.5, pp.1–8 (2009).
7) 伊藤祥司，杉原正顯：線形方程式求解に対するクリロフ部分空間法の前処理系に着目した体系的な特性分析，日本応用数理学会 2009 年度年会講演予稿集，大阪 (Sep. 2009).
8) Lis. http://www.ssisc.org/lis/
9) Matlab コード（Netlib）. http://www.netlib.org/templates/matlab/
10) Matrix Market. http://math.nist.gov/MatrixMarket/
11) 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).
12) Saad, Y.: *Iterative Methods for Sparse Linear Systems, 2nd ed.*, SIAM (2003).
13) Sonneveld, P.: CGS, A fast Lanczos-type solver for nonsymmetric linear systems, *SIAM J. Sci. Stat. Comput.*, Vol.10, No.1, pp.36–52 (1989).
14) Van der Vorst, Henk A.: BI-CGSTAB: A fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems, *SIAM J. Sci. Stat. Comput.*, Vol.13, No.2, pp.631–644 (1992).
15) Van der Vorst, Henk A.: *Iterative Krylov Methods for Large Linear Systems*, Cambridge University Press (2003).

(平成 21 年 9 月 29 日受付)
(平成 22 年 3 月 22 日採録)

伊藤　祥司（正会員）
2001 年 3 月筑波大学大学院工学研究科博士課程修了．博士（工学）．筑波大学大学院システム情報工学研究科，理化学研究所を経て，現在，東京大学情報基盤センター特任准教授．数値計算アルゴリズムの体系的性能評価，求解プロセスに対する品質管理の研究に従事．日本応用数理学会，SIAM，ACM 各会員．

杉原　正顯（正会員）
1982 年 3 月東京大学大学院工学系研究科博士課程修了．工学博士．現在，東京大学大学院情報理工学系研究科教授．数値計算の基本アルゴリズムの解析と設計の研究に従事．日本応用数理学会，日本数学会各会員．

姫野龍太郎（正会員）
1981 年 3 月東京大学大学院工学系研究科博士課程修了．工学博士．現在，独立行政法人理化学研究所情報基盤センター長．数値流体計算，数値計算の基本アルゴリズムの設計の研究に従事．日本応用数理学会，日本計算工学会各会員．