2008－HPC－116 (20)
2008／8／6

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

伊藤祥司
理化学研究所 情報基盤センター

線形方程式を例に取り上げて，求解アルゴリズムの体系的な性能比較の方法を提案する．この中で紹介する情報システムでは，典型的なテスト問題を用いて反復解法と前処理に対する求解性能情報のDBを生成し，線形方程式と求解アルゴリズムの間の性能評価を可視的に行う．本稿では特に，求解アルゴリズムの中でスケーリングも含めて前処理の効果を比較した．さらに，前処理を施したときの求解アルゴリズムにおける残差ベクトルの違いに着目して，スケーリングを用いる際の問題点について指摘した．

Effective performance of preconditioning technique for solving linear equations from a viewpoint of systematic evaluation

Shoji ITOH
Advanced Center for Computing and Communication, RIKEN

Systematical performance evaluation methods for comparing numerical algorithms for linear equations are proposed. An introduced computational system generates a performance information database for a range of iterative solvers and preconditioning using a lot of typical test problems, and presents data visually allowing the relationships between the various algorithms and problems to be compared. In this paper, especially, effects of scaling and preconditioning are discussed. Further, residual vector in solution algorithms is paid attention to and some related issues 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×n の係数行列を持つ線形方程式

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

を解くことに帰着され，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（解法および解法と併用する技法）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]．従って，実際の求解問題に対して，どの求解アルゴリズムを適用したら良いか指針が欲しい．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を行ってきており，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，このようにして得た情報を自動チューニングへ応用する場合，求解アルゴリズムの選択機構などでも利用できる．そのとき，適切なアルゴリズムを選択すること以外に，不適切なアルゴリズムを選択しない方法なども考えられる．

本稿では，前処理の効果に注目して，反復解法に様々な前処理を適用したときの求解効率の比較結果について報告する．

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）[4] の逐次版を使用した．バージョンはlis-1.1.2修正版である．本稿の中では，Krylov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[5]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$\|r_k\|_2/\|b\|_2 \leqq 1.0\times10^{-12}$を満たしたときとした（$r_k$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に記載されているデフォルトを用いた．

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，ジョブ実行してデータ採取することである．

3. 求解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．

スケーリング：係数行列の値の大きさを揃え，数値計算上の安定性を向上させる．スケーリングにより係数行列の条件数も改善される．その演算方法は，式(2)のとおり元の係数行列の対角要素の逆数から構成される行列を施す方法の他に，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．行列DをAの対角要素により構成したときは，

$$D^{-1}Ax = D^{-1}b \tag{2}$$

である．

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．

例えば，$A \approx K$となる前処理行列Kを用いて，

$$K^{-1}Ax = K^{-1}b \tag{3}$$

と式(1)を変換する．このような前処理を施すことで，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．実際には，式(3)自体は作らず，こ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．

オーダリング：ガウス消去法に基づく前処理演算など演算過程でfill-in発生の減少や，演算回数を減らすような合理化を図るため元の係数行列の順序を入れ替える技法である．ベクトル並列計算などの再帰演算を回避する方法としても重要であるが本稿の中では取り上げない．

本稿では，スケーリングと狭義の前処理がもたらす効果について評価する．これら技法の理論面からの考察では，係数行列の条件数や固有値分布など行列のスペクトル特性に基づいた議論がなされることが多いが，実際の数値計算では，計算機が表現できる精度の限界（例えば，倍精度演算などによる表現範囲）をはじめ，様々な原因により，得られた求解性能を示す情報からは，必ずしも理論の正しさを示す結果とはならない場合も多い．更には，不完全分解系の前処理の効果などは，理論に基づいて定量化することが難しい．従って，実際に前処理付きアルゴリズムを用いて線形方程式を解き，各々の求解性能を示すデータを比較することで，前処理の効果を比較評価する．

式(3)で前処理行列KをAの対角要素により構成したときは，式(2)のスケーリングと一見同じに見えるが，実際には求解アルゴリズムのレベルで幾つか異なる点がある．スケーリングでは，予め(2)の演算を行い，線形方程式そのものを，

$$\breve{A}x = \breve{b}, \tag{4}$$
$$\breve{A} = D^{-1}A, \tag{5}$$
$$\breve{b} = D^{-1}b \tag{6}$$

と変形する．つまり，求解アルゴリズムに入力する情報（係数行列，右辺項）は，(4)のとおり変形された後の系の情報である．一方，前処理では，求解アルゴリズムの中で，(3)と同値となる前処理演算を付加する．そのときの“前処理付きアルゴリズム”に入力する情報は，(1)で表された係数行列と右辺項のままである．スケーリングを用いたときの残差ベクトルは，

$$\breve{b} - \breve{A}x_k \tag{7}$$

であり，(3)に基づく前処理付きアルゴリズムでは，

$$K^{-1}(b - Ax_k) \tag{8}$$

で表される．本来評価すべき残差ベクトルは，

$$b - Ax_k \tag{9}$$

なので，(7),(8)は共に本来とは異なる系で残差ベクトルが構成されている．これについては4.2節でも議論する．

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提案し，それを用いて新たに確認できる事象について説明する．

4. 1 絶対的な指標を用いたスケーリング有無の2つのSurvey Chartの比較

これまでの先行研究では，１つの線形方程式について，Lisの全アルゴリズムを用いて得られた求解結果に対し，収束までの所要反復回数が最小であったアルゴリズムと，それ以外のアルゴリズムとの比を算出し，その数値を指標として性能を比較してきた[3]．つまり，各々の求解アルゴリズムに対し，

Score = 10 × (最小の所要反復回数
÷ それ以外のアルゴリズムの所要反復回数) (10)

を算出した．

今回は，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．

$$\text{Score} = 10 - \left[\frac{\text{所要反復回数} - 1}{\text{係数行列のサイズ}} \times 10 \right] \tag{11}$$

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，所要反復回数の数値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．より厳しく評価するためには，20等分，50等分などにクラス分けすれば良い（図１）．

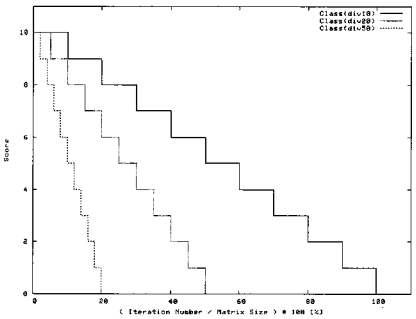

図１ 絶対的指標を表す評価式のグラフ

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，式(11)の方針に基づき50等分にクラス分けして上位10クラスに対してScoreを付した（図１の最左のグラフ）．この値に応じて，図２のようにセル（小枠）を色分けして図示したものが図３，４である．

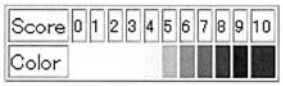

図２ Scoreと配色の対応

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（実際のシステムでは色による識別が可能だが，本稿ではモノクロのため，グレースケールの濃淡による表現となっている）．

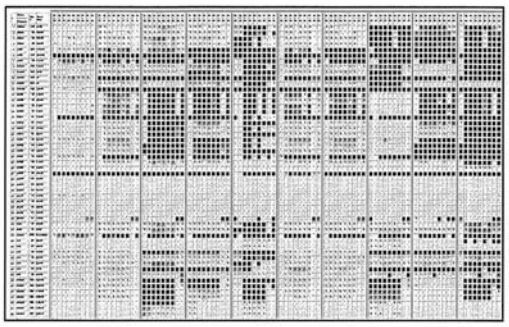

図3 前処理でグループ分けしたSurvey Chart
（スケーリング無し）

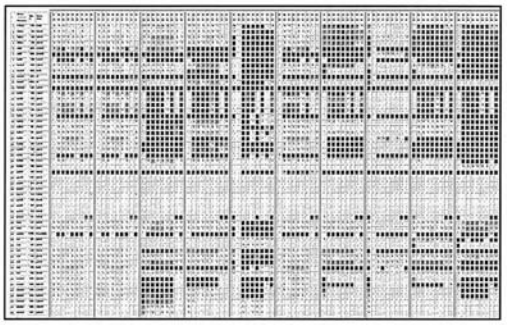

図4 前処理でグループ分けしたSurvey Chart
（スケーリング有り）

また，「全く収束していない」場合にはドットを記している．式(12)による真の残差ベクトルの相対残差ノルムが10^{-8}以上であった「見かけ上の収束」の場合にはアスタリスクを記している．

$$\frac{\|\hat{r}\|_2}{\|b\|_2} = \frac{\|b - A\hat{x}\|_2}{\|b\|_2} \qquad (12)$$

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．図３，４では，図中のフレーム（太い枠）により10種類の前処理グループを分類しており，各々のフレームの中に各解法の結果が並んでいる．図３，４で前処理と解法の並び順は同一であり，2節に記載した順序と同じである．先行研究での報告では，アルゴリズムの体系的な性能比較の結果，反復解法よりも前処理の方が収束に及ぼす効果が大きいことを確認している[3]．したがって，今回の報告では前処理の効果を比較する結果のみを示す．

図３の最左のフレームは「前処理無し」の結果であり，それに対応する図４のフレームは「前処理を施さずスケーリングのみ」の結果である．この２つからスケーリングのみを併用したときの収束性に対する効果を確認できる．

図３，４の比較からスケーリング自体の効果は確認できるが，それは点Jacobi前処理とほぼ同等の性能である．また，本実験条件の下では，スケーリングと前処理とを併用した場合でも，収束状況が劇的に変わるようなことは無い．収束状況に変化のあるケースも多少はあるが，それは，併用する前処理の種類，線形方程式の問題に応じて結果の傾向も異なる（つまり，個別の話題として議論すべきである事柄である）．一方で，スケーリング無しでは収束しているが，スケーリングありでは全く収束しなくなるケースも確認された．

4. 2 左前処理と右前処理の違いによる収束状況の違い

ところで，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とき，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“見かけ上の収束”となる結果が多く確認された．ここでは，スケーリングを施さずに求解した計算結果であったのだが，スケーリングを施した場合には，その件数が減少した．結果だけを見るとスケーリングの効果として考えられないことも無いが，ここでは，もう少し踏み込んで分析した．

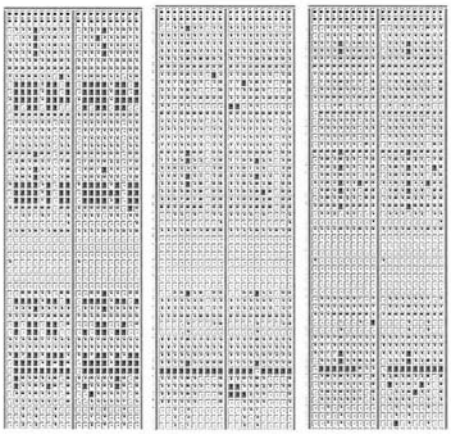

図5 左：スケーリング無し左前処理，中：スケーリング付き左前処理，右：スケーリング無し右前処理

図５は，上述の結果が確認された２種類の解法をグループ分けして，全前処理との組合せの結果を示したものである．ここでは，見かけ上の収束をグレーで表示し，Scoreの値に対しては着色していない．すなわち，図２のColorはすべて白である（これも，実際のシステムでは多くの色による識別を可能にしているが，本稿ではモノクロのため，このように表現した）．スケーリング無しで求解したときに見かけ上の収束が多かった解法（図５左）では，左前処理が用いられていた．これらを右前処理に変更して計算したところ，スケーリング無しの場合でも，見かけ上の収束が減少した（図５右）．

ここで，左右前処理などの前処理方法について概説する．式(3)で表した前処理は，線形方程式(1)の左側から前処理を施したものであり，左前処理と呼ばれる．その他には，右前処理：

$$AK^{-1}Kx = b. \tag{13}$$

更に，前処理行列を $K = K_1 K_2$ と分解したときの両側前処理：

$$K_1^{-1} A K_2^{-1} K_2 x = K_1^{-1} b \tag{14}$$

がある．特に両側前処理は，係数行列が対称行列の場合，その対称性を保持することを目的として用いられる場合が多い．そして，これら３種類の前処理では，記述される前処理付きアルゴリズムも異なる．Y. Saadの文献[6]では，GMRES法の左前処理と右前処理の場合のアルゴリズムが記載されている．

これら３種類の前処理が施された系の中で，残差ベクトルが本来の

$$b - Ax_k \tag{15}$$

として表されるのは，右前処理の系(13)のみである[6,7]．

以上から，図５左のスケーリング無し左前処理の場合に見かけ上の収束が多かった一因として，求解アルゴリズム中の残差ベクトルが本来評価すべき情報とは異なっていたことが考えられる．また，図５中ではスケーリングの併用により左前処理でもほとんどの問題が収束してはいるが，この場合，求解アルゴリズムの合理性という点では，本質的な問題は解決されていないことを留意しておくべきである．

5. 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った．求解アルゴリズムの前処理，特にスケーリングの影響に着目した．

本稿の中では，反復法に基づく求解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．狭義の前処理にスケーリングを併用した場合としない場合との収束状況について確認したところ，併用がもたらす効果とは一般的なものではなく，特定の前処理，特定の問題で認められるものであった．中には，スケーリングの併用により全く収束しなくなった例も少なからず確認された．

また，体系的性能評価の中で特定の解法で多く見られた“見かけ上の収束”に焦点を当てた．一因として，それらは求解アルゴリズム中で表されている残差ベクトルの違いが挙げられ，狭義前処理の実装における問題であることと，スケーリングを併用して収束させた場合の問題点について指摘した．

謝辞：本研究は理化学研究所「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成果である．

参考文献

[1] Barrett, R., et. al., *Templates for the solution of linear systems: Building Blocks for Iterative Methods*, SIAM, 1994.（邦訳）長谷川里美，長谷川秀彦，藤野清次：反復法 Templates，朝倉書店，1996.

[2] Itoh, S., et. al., 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，数値計算アルゴリズム性能情報 DB の自動チューニング技術への適用の検討，情報処理学会 研究報告, 2007-HPC-111, pp.201-206, (2007).

[4] Lis, http://www.ssisc.org/lis/

[5] Matrix Market, http://math.nist.gov/MatrixMarket/

[6] Saad, Y., *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company, 1996.

[7] Van der Vorst, A. H., *Iterative Krylov Methods for Large Linear Systems*, Cambridge University Press, 2003.