クリロフ部分空間法に対する多倍長演算の効果

長谷川秀彦* 阿部邦美†
*図書館情報大学 †理化学研究所

1 はじめに

共役勾配法（Conjugate Gradient Method，CG 法）に代表されるクリロフ部分空間法は，大規模で疎な連立一次方程式 $\boldsymbol{Ax} = \boldsymbol{b}$ を効率良く解くことができる．しかし，問題の条件が悪くなるにしたがって数値誤差の影響を受け，解に到達しないこともある．また，理論的に収束が保証されていても，コンピュータを用いた有限桁の数値演算ではその性質は失われる．さらに，計算機代数のように厳密計算を行うことはできるが，多くの場合，入力として与えられる数値にも誤差が入っており，膨大なコストを支払って計算過程だけを厳密に計算しても解に収束するとは限らない．

実用的には前処理を行ない，係数行列を効率的に解けるように変換してから，クリロフ部分空間法を適用する．その代表的な前処理には不完全コレスキー分解 (Incomplete cholesky factorization, IC)（対称の場合）[3] や不完全 LU 分解 (Incomplete LU factorization, ILU)（非対称の場合）[3, 4] があり，これらはクリロフ部分空間法の反復回数を大幅に減少させることが知られている．ところが，これらは本質的に逐次演算であり，ベクトルコンピュータや分散メモリ方式の並列コンピュータでは実装がむずかしい．

そこで，われわれは多倍長演算によって反復回数の減少をはかる．多倍長演算は演算コストが高い一方で，ベクトルコンピュータや並列コンピュータでは大容量のメモリと高速演算が可能である．それゆえ，うまくいけばスムーズな収束特性と，より良い解が得られると考えられる．

2 前処理付きクリロフ部分空間法

前処理付きクリロフ部分空間法の収束特性に影響を与える条件を以下に列挙する．

- 問題に内在する性質（条件数，固有値分布，固有ベクトル成分）
- 初期値
- 前処理
- 解法
- 演算精度（丸め誤差）
- 計算環境

問題に内在する性質は解法レベルでは対策できないが，解法全体に大きな影響を与えている．また，よい初期値が与えられれば収束を改善できるが，一般的には困難である．次に，数学的に良い前処理，すなわち不完全 LU 分解や近似逆行列を用いる場合，反復回数を大幅に減少させることができるが，逐次処理を含むため並列コンピュータ上ではコスト高になる．一方，並列コンピュータを使用する観点から好ましい前処理，すなわちスケーリングやマルチカラー法を用いる場合，並列度が高く並列コンピュータの性能をフルに引き出すことができるが，大きな反復回数の減少は望めない．また数学的

には，並列度をあげるために一部の依存関係を完全に無視して計算している．したがって，問題が難しくなるほど，解を得るためには数学的に良い前処理を使う必要が生じ，結果的に並列コンピュータの性能をフルには活用できない事態もありうる．計算環境と問題の組合せを考えたとき，完璧な前処理は存在しない．

クリロフ部分空間法のアルゴリズムは，行列・ベクトル積，内積，ベクトルの和，ベクトルのスカラー倍などからなり，並列性を内在している．また，多倍長演算を行なう場合はこれらの計算に内在する並列性を損なわないので，多倍長演算を導入すること，例えば倍精度演算を 4 倍精度演算にすることによって，自動的により良い解を得ることが可能となる．PC，ワークステーション，SMP，ベクトルコンピュータ，分散メモリ方式並列コンピュータなどの計算環境 [5] によって，高速なアルゴリズムは異なる．これらの環境全体に言えることは，メモリの大容量化のペースは速い一方で， CPU の演算速度の向上に対してメモリアクセススピードの向上が追随していないことである．そのため，同じ演算量でもメモリ参照回数の少ないアルゴリズムの方が早く結果を得られる．また，特に大規模なメモリが搭載可能な SMP や分散メモリ方式の並列コンピュータでは，メモリ容量は制約になりにくい．

3　数値実験

3.1　モデル問題と手法

前提は，クリロフ部分空間法を用いてベクトルコンピュータや分散メモリ方式並列コンピュータ上で大規模で疎な連立一次方程式を高速に解くことである．われわれは倍精度演算で前処理をしない場合，倍精度演算で不完全 LU 分解によって前処理を行なう場合，4 倍精度で前処理をしない場合の 3 種類を比較する．IEEE Standards 754 方式のハードウェアを実装した PC やワークステーションの場合，倍精度演算はハードウェアで実行されるが，4 倍精度演算はエミュレーションで実行されるため，現状では優位性がほとんどない．

モデル問題として Gutknecht が用いた 200 次元の Toeplitz 行列 [2] を利用した．パラメータは $\gamma = 1.3, 1.7, 2.1, 2.5$ と変化させた．このとき，パラメータの値を増加させるとクリロフ部分空間法は収束しにくくなる．また右辺項は $\boldsymbol{b} = (1, 1, \ldots, 1)^{\mathrm{T}}$ とする．

$$A := \begin{bmatrix} 2 & 1 & & & \\ 0 & 2 & 1 & & \\ \gamma & 0 & 2 & 1 & \\ & \gamma & 0 & 2 & \ddots \\ & & \ddots & \ddots & \ddots \end{bmatrix}$$

解法は Templates[1] の Bi-Conjugate Gradient Method（双共役勾配法，Bi-CG 法）（Fortran 版）を用いた．Bi-CG 法を用いた理由は，行列・ベクトル積，転置行列・ベクトル積があり，演算の種類による影響も考察できると考えたためである．初期値は $\boldsymbol{x}_0 = 0$, 停止条件は $\|\boldsymbol{r}_k\|/\|\boldsymbol{r}_0\| = 10^{-12}$ を採用した．また，解法の最大反復回数は 2000 回とした．

3.2　結果と考察

所要反復回数，計算時間，相対残差ノルム $\|\boldsymbol{b} - A\boldsymbol{x}_k\|_2 / \|\boldsymbol{b}\|_2$（$\boldsymbol{x}_k$ はアルゴリズムで漸化的に計算された近似解である）を Table 1に示す．

次に，Bi-CG 法における 1 回反復当たりの行列・ベクトル積，転置行列・ベクトル積，ベクトル和，内積，ノルムなどに関する計算時間の割合を Table 2に示す．

また，IBM SP2 で並列実行した際の時間短縮比率を Table 3[6] に示す.

		$\gamma = 1.3$	$\gamma = 1.7$	$\gamma = 2.1$	$\gamma = 2.5$
倍精度 + 前処理なし	Iter.	188			
	Time	1.7×10^{-1}	×	×	×
	Residual	3.1×10^{-13}			
倍精度 + ILU 前処理	Iter.	40	75	148	179
	Time	6.8×10^{-2}	8.5×10^{-2}	1.1×10^{-2}	1.4×10^{-2}
	Residual	1.7×10^{-13}	5.6×10^{-13}	8.3×10^{-10}	5.2×10^{-2}
4 倍精度+ 前処理なし	Iter.	122	152	255	446
	Time	2.0×10^{2}	5.3×10^{2}	5.5×10^{3}	1.9×10^{5}
	Residual	6.6×10^{-13}	6.4×10^{-13}	3.6×10^{-13}	5.4×10^{-13}

Table. 1. Toeplitz 行列に対する収束状況（Sun Enterprize 3000 ($n = 200$)）

[**収束に関する考察**]

不完全 LU 分解前処理なしの倍精度演算の場合，$\gamma = 1.7, 2.1, 2.5$ のとき収束しなかった．さらに，$\gamma = 2.5$ の場合，倍精度演算で不完全 LU 分解前処理を施したときには 179 回で停止している．しかし，相対残差ノルム $\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k\|_2 / \|\boldsymbol{b}\|_2$ は 10^{-2} で，十分な精度の解は得られていない．また，$\gamma = 2.1$ の場合も，倍精度演算で不完全 LU 分解前処理を施したときには所要反復回数が 148 回で，そのときの残差ノルム $\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k\|_2 / \|\boldsymbol{b}\|_2$ は 10^{-10} であった．したがって，倍精度では解けない問題は珍しくないことがわかる．

	Sun Enterprize 3000($n = 200$)		Fujitsu VPP300($n = 20000$)	
	Double	Quadruple	Double	Quadruple
Time/iter.	$1.27 * 10^{-3}$s	$5.71 * 10^{-2}$s	$3.41 * 10^{-2}$s	6.63s
MATVEC	5%	7%	1%	10%
MATVECT	5%	7%	1%	10%
DAXPY DCOPY DDOT DNORM2	80%	74%	28%	50%
PSOLVE(ILU)	5%	6%	35%	15%
PSOLVET(ILU)	5%	6%	35%	15%

Table. 2. 1 反復当たりの実行時間と構成比

[**コストに関する考察**]

ワークステーション Sun Enterprize 3000 の 4 倍精度演算の実行時間は倍精度演算の 45 倍，ベクトルコンピュータ Fujitsu VPP300 の場合は 194 倍である．ワークステーション上では行列・ベクトル積と不完全 LU 分解前処理の演算量はほぼ同程度であるが，ベクトルコンピュータの場合，前処理に行列・ベクトル積の 35 倍の時間がかかっている（超平面法は使用していない）．これは行列・ベクトル積がベクトル演算で実行された効果である．一方，4 倍精度演算では，行列・ベクトル積と前処

	1 PE	2 PE	4 PE	8 PE	16 PE
MATVEC	1	0.61	0.38	0.26	0.20*1
MATVECT	1	0.61	0.38	0.26	0.20*1
DAXPY DCOPY DDOT DNORM2	1	0.57	0.38	0.34	0.32*1
PSOLVE(ILU)	1	1.1	0.54	0.36	0.34*2
PSOLVET(ILU)	1	1.1	0.54	0.36	0.34*2

Table. 3. 並列化による時間短縮率（IBM SP2, *1: $n = 40000$, *2: $n = 19200$ ）

理に大きな差はない．これらのことから，この計算環境では４倍精度演算には優位性が現れないことが予想される．次に，IBM SP2 上の倍精度演算では，並列化したとき，行列・ベクトル積は 16 台で 5 倍程度，不完全 LU 前処理は 3 倍程度にしか高速化されないことを示している．

4 まとめ

不完全 LU 分解前処理は数学的に強力な前処理であり，反復回数を大幅に削減できるが，ベクトルコンピュータや分散メモリ方式並列コンピュータでは演算コストが高い．一方，４倍精度演算は収束を安定確実にできるが，現状では非常にコストがかかる．しかし，問題によっては，倍精度演算で不完全 LU 分解前処理を施した場合より４倍精度演算による計算の方が有効な場合がある．さらに，４倍精度演算の場合には今回は前処理を施していないが，並列化が可能なスケーリングなどの前処理を施せばさらに効果が期待できる．したがって，分散メモリ方式並列コンピュータ上で並列化可能な前処理と４倍精度演算を組み合わせることによって，不完全 LU 分解前処理を施した場合より優位性を引き出せる可能性をもつ．結果は会場で報告する．

参考文献

[1] Barrett, R., Berry, M., Chan, T., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., and van der Vorst, H., *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, 1994. 邦訳（長谷川里見, 長谷川秀彦, 藤野清次）, 反復法 Templates, 朝倉書店, 東京, 1996.

[2] Gutknecht, M. H., Variants of BiCGSTAB for Matrix with Complex Spectrum, *SIAM J. Sci. Comput.*, **14** (1993), 1020-1033.

[3] Meijerink, J. A. and van der Vorst, H. A., An Iterative Solution Method for Linear Systems of which the Coefficient Matrix is a Symmetric M-matrix, *Mathematics of Computation*, **31** (1977), 148-162.

[4] Saad, Y., *Iterative Methods for Sparse Linear Systems*, PWS, Boston, 1996.

[5] 湯浅太一, 安村通晃, 中田登志之 編, はじめての並列プログラミング, 共立出版, 東京, 1998.

[6] 私信, 日本原始力研究所より.