

非対称行列から生成された対称行列に対するCG法

筑波大学 図書館情報学系 長谷川秀彦

Hidehiko Hasegawa, Inst. of Library and Information Science, University of Tsukuba

東京大学大学院 工学系研究科 曾我部知広

Tomohiro Sogabe, School of Engineering, The University of Tokyo

早稲田大学 理工学研究科 荻田武史

Takeshi Ogita, Waseda University

1 はじめに

A を $N \times N$ の非対称疎行列, b を N 次元ベクトルとした大規模な線形方程式

$$Ax = b \quad (1)$$

に対しては, BiCG 法, GMRES 法, GPBiCG 法などの非対称行列用の Krylov 部分空間法が使われる。しかし, 対称正定値行列向けの Conjugate Gradient (CG) 法に比べると, 非対称行列用の反復解法は収束が安定しにくい。

$$A^T Ax = A^T b \quad (2)$$

とすれば, 対称正定値行列 $A^T A$ を係数とする方程式になって CG 法が適用できるが, $A^T A$ の条件数は A の 2 乗になるので精度の点で好ましくない。

本研究では, 大規模疎行列を前提に, 条件数を変えずに対称化する方法を検討し, 数値実験により対称化した方程式に CG 法を適用したときの収束特性を明らかにする。仮に, 対称化によって次元数が 2 倍になっても, 反復回数が同程度かそれ以下であれば, 対称性を利用したアルゴリズムを構成することで非対称行列に対する解法よりも高速になりうる。ただし, 実際のスピードは計算環境とアルゴリズムの実現方式に強く依存する。

2 対称化の方法

非対称 Toeplitz 行列に対しては

$$P = \begin{pmatrix} & & 1 \\ & \dots & \\ 1 & & \end{pmatrix} = P^T$$

を用いれば, 条件数を変えずに対称化でき [1], 対称化した方程式

$$PAx = Pb, \quad (3)$$

$$AP(P^{-1}x) = b \quad (4)$$

が得られる。このような A, PA を係数とする方程式 (1), (3) に対して CG, BiCG, BiCGSTAB, QMR, CGS, GMRES, LSQR などを適用した結果, PA による収束性の改善は問題依存であることが示されている [2]。

類似の方法で、一般の $N \times N$ の非対称行列 A に対して、条件数を変えずに次元数が $2N$ の対称化した方程式

$$\begin{pmatrix} & A^T \\ A & \end{pmatrix} \begin{pmatrix} x \\ o \end{pmatrix} = \begin{pmatrix} o \\ b \end{pmatrix} \quad (5)$$

を作り、対称正定値行列用の解法である CG 法を適用する。しかし、この方程式で初期解 $x_0 = o$ とおくと、後述の理由により最初の反復で破綻が生じる。

そこで、この方程式を一般化して条件数が同じ $2N$ 次元の対称化した方程式

$$\begin{pmatrix} & A^T \\ A & \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} b' \\ b \end{pmatrix}, \quad (6)$$

$$\begin{pmatrix} & A \\ A^T & \end{pmatrix} \begin{pmatrix} y \\ x \end{pmatrix} = \begin{pmatrix} b \\ b' \end{pmatrix} \quad (7)$$

を作る。右辺 b' と対応する解 y にはいくつかの選択肢が考えられ、それによって $2N$ 次元の方程式の性質が変化する。係数行列が正定値なら CG 法の収束が理論的に保証されるが、正定値でなくとも CG 法は多くの場合に収束する。さらに、丸め誤差の影響によって収束しない場合は、多倍長演算などを使えば収束が改善される [3]。

いっぽう、(6), (7) に対角成分を加えて一般化したのが Ayachour[4], Saunders[5],[6] らの方程式で、 λ, δ を変化させた反復計算により目的の解を得る。方程式 (6) は $\lambda x + A^T x' = b'$ に対して $\lambda \rightarrow 0$ と収束させたものになっている。係数行列の条件数は変わるが、対角要素を人為的に補うことにより不完全コレスキー分解などの前処理が可能になる。この場合も、 b', c の与え方には選択の余地がある。

$$\begin{pmatrix} \lambda I & A^T \\ A & O \end{pmatrix} \begin{pmatrix} x \\ x' \end{pmatrix} = \begin{pmatrix} b' \\ b \end{pmatrix}, \quad (8)$$

$$\begin{pmatrix} \delta I & A \\ A^T & -\delta I \end{pmatrix} \begin{pmatrix} s \\ x \end{pmatrix} = \begin{pmatrix} b \\ c \end{pmatrix}. \quad (9)$$

いっぽう、荻田は条件数が同じ行列を係数に持つ方程式

$$\begin{pmatrix} A + A^T & A - A^T \\ -(A - A^T) & -(A + A^T) \end{pmatrix} \begin{pmatrix} x \\ x \end{pmatrix} = 2 \begin{pmatrix} b \\ -b \end{pmatrix} \quad (10)$$

を提案している [7]。この方程式には任意性のある項・パラメータが存在せず、係数行列の疎度（非ゼロ要素の率）は最大で 4 倍、差分法などで離散化して得られた非ゼロ要素が対称の位置に現れる行列の場合には不変という特徴を持つ。

2.1 アルゴリズム

方程式 (6) に対する CG 法のアルゴリズムを図 1 に示す。アルゴリズム中の x, b', r', p' は上半分のベクトルを意味する。 α_k, β_k の計算で上半分のベクトルと下半分のベクトルが同時に使われ、その後のステップで上半分と下半分の情報が交換される。図 2 の BiCG 法のアルゴリズム [8] とは、 α_k, β_k の計算方法だけが異なるだけで、アルゴリズム全体の構造は BiCG 法と同様になり、演算量も BiCG 法と同じになる。並列化の条件についてもほぼ同等である。

BiCG 法との大きな違いは、 $2N$ 次元の係数行列を持つ全体の系に対してアルゴリズムどおりに収束判定をすると $\|r\| + \|r'\|$ を使うことである。これは、 $\|r\|$ で収束判定を行う N 次元の問題に比べると厳しい。アルゴリズム全体の進行には $\|r\| + \|r'\|$ の値が必要だが、解 x に対する収束判定だけなら $\|r\|$ を用いるべきだろう。 b' を構成する際には、最初に適当な y を考え、そこから $b' = A^T y$ を作り、初期値として y を使うことも不可能ではない。この場合、すでに解の定まった N 次元の方程式と、解に対する情報がなにもない N 次元の方程式を連立させて解くことになるが、このようなアンバランスが全体の収束にどのような影響を及ぼすかについても興味がある。

$$\begin{aligned}
 & \text{初期ベクトル } x_0, y_0 \text{ を用意} \\
 & r_0 = b - Ax_0; \quad r'_0 = b' - A^T y_0 \\
 & p_0 = r_0, \quad p'_0 = r'_0 \\
 & \text{for } k = 0, 1, \dots \\
 & \quad \alpha_k = \frac{[(r_k, r_k) + (r'_k, r'_k)]}{[(p_k, Ap'_k) + (p'_k, A^T p_k)]} \quad \left\{ = \frac{\|r_k\|^2 + \|r'_k\|^2}{2(p_k, Ap'_k)} \right\} \\
 & \quad x_{k+1} = x_k + \alpha_k p'_k \quad \{; \quad y_{k+1} = y_k + \alpha_k p_k \} \\
 & \quad r_{k+1} = r_k - \alpha_k Ap'_k; \quad r'_{k+1} = r'_k - \alpha_k A^T p_k \\
 & \quad \text{収束判定} \\
 & \quad \beta_k = \frac{[(r_{k+1}, r_{k+1}) + (r'_{k+1}, r'_{k+1})]}{[(r_k, r_k) + (r'_k, r'_k)]} \\
 & \quad p_{k+1} = r_{k+1} + \beta_k p_k; \quad p'_{k+1} = r'_{k+1} + \beta_k p'_k
 \end{aligned}$$

図 1: 対称化した方程式に対する CG 法のアルゴリズム

$$\begin{aligned}
 & \text{初期ベクトル } x_0, \tilde{x}_0 \text{ を用意} \\
 & r_0 = b - Ax_0; \quad \tilde{r}_0 = \tilde{b} - A^T \tilde{x}_0 \\
 & p_0 = r_0; \quad \tilde{p}_0 = \tilde{r}_0 \\
 & \text{for } k = 0, 1, \dots \\
 & \quad \alpha_k = \frac{(r_k, \tilde{r}_k)}{(Ap_k, \tilde{p}_k)} \\
 & \quad x_{k+1} = x_k + \alpha_k p_k \\
 & \quad r_{k+1} = r_k - \alpha_k Ap_k; \quad \tilde{r}_{k+1} = \tilde{r}_k - \alpha_k A^T \tilde{p}_k \\
 & \quad \text{収束判定} \\
 & \quad \beta_k = \frac{(r_{k+1}, \tilde{r}_{k+1})}{(r_k, \tilde{r}_k)} \\
 & \quad p_{k+1} = r_{k+1} + \beta_k p_k; \quad \tilde{p}_{k+1} = \tilde{r}_{k+1} + \beta_k \tilde{p}_k
 \end{aligned}$$

図 2: 非対称行列 A に対する BiCG 法のアルゴリズム

2.2 前処理

CG 法の収束を改善するには、なんらかの前処理が必要である。(6), (7) は対角要素が存在せず、対角スケーリング、不完全コレスキー分解などの前処理は困難だが、たとえば一般的な対称行列に適用可能な近似逆行列を用いた前処理などが適用できるだろう [9]。

一方、拡大行列を構成する行列 A に対しては、一般的な非対称行列用の効率的な前処理が適用できる。たとえば、不完全 LU 分解を適用するのに必要となるのは $(LDU)^{-1}$, $(LDU)^{-T}$ だが、これらは非対称行列 A に対する前処理がそのまま使える。ここで $\mathbf{y}' = (LDU)^T \mathbf{y}$ である。

$$\begin{pmatrix} A^T(LDU)^{-T} \\ (LDU)^{-1}A \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{y}' \end{pmatrix} = \begin{pmatrix} \mathbf{b}' \\ (LDU)^{-1}\mathbf{b} \end{pmatrix}, \quad (11)$$

$$\begin{pmatrix} A^T(LDU)^{-T} & (LDU)^{-1}A \end{pmatrix} \begin{pmatrix} \mathbf{y}' \\ \mathbf{x} \end{pmatrix} = \begin{pmatrix} (LDU)^{-1}\mathbf{b} \\ \mathbf{b}' \end{pmatrix}. \quad (12)$$

効果的な前処理は係数行列の性質、右辺、初期値に依存するが、この二通りの前処理は併用できる。また、拡大行列を係数とする方程式を構成する個々の方程式に対する前処理が全体の系にとっていいのか悪いのかは実験&解析がいる。

3 数値実験

ここでは A として直方体の場における移流拡散方程式を差分法 (Control Volume 法) で離散化して得られた行列を扱う [10]。セルペクレ数 ($v/5$) が 2 以下の場合には、対角優位行列となり、不完全 LU 分解を前処理に用いた BiCG 法などに適した問題となる。実験には、分割を $5 \times 5 \times 10$ とした $N = 250$ の問題を用いる。

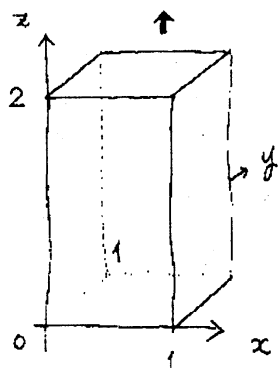
方程式 (6), (7) の $\mathbf{b}'$ として現実的に使えるものは、 $\mathbf{b}$, $P\mathbf{b}$, $(1, 0, \dots, 0)^T$, $(1, 1, \dots, 1)^T$, 乱数, 乱数 $\cdot \|\mathbf{b}\|$ などである。そのほか、 $\mathbf{y}$ を適当に定めて $\mathbf{b}' = A^T \mathbf{y}$ で作ることもできる。しかし、図 1 のアルゴリズムで $\mathbf{b}' = (0, 0, \dots, 0)^T$ のとき、初期解 $\mathbf{y}_0 = (0, 0, \dots, 0)^T$ とすると、 $\mathbf{p}'_0 = \mathbf{r}'_0 = \mathbf{b}' - A^T \mathbf{y}_0$ より $\mathbf{p}'_0 = \mathbf{o}$ となって α_0 の計算で破綻が生じる。図 2 の BiCG 法でも $\tilde{\mathbf{x}}_0 = \tilde{\mathbf{b}} = \mathbf{o}$ として、 $\tilde{\mathbf{p}}_0 = \tilde{\mathbf{r}}_0$ を用いたときには同様の破綻が生じる。

$A\mathbf{x} = \mathbf{b}$ を BiCG で解いた場合の反復回数と対称化した方程式 (7) の $\mathbf{b}'$ を変えて CG 法で解いたときの反復回数を表 1 に示す。収束判定は相対残差ノルムが 10^{-10} 以下とし、初期解は $\mathbf{o}$ とした。乱数は $[0, 1]$ の一様乱数である。

表 1: 右辺 $\mathbf{b}'$ とセルペクレ数への依存性 (反復回数)

	0(対称)	1	2	4	8	16	32	64
BiCG	46	52	47	54	71	89	116	140
$\mathbf{b}$	47	278	250	212	182	169	188	210
$P\mathbf{b}$	358	329	290	246	210	198	216	250
$(1, 0, \dots, 0)^T$	238	332	296	246	208	200	218	258
$(1, 1, \dots, 1)^T$	230	330	294	242	208	194	216	250
乱数 $\cdot \ \mathbf{b}\ $	398	382	332	276	236	216	238	262
乱数	500	496	410	324	252	242	268	318

$$-\operatorname{div}(k \nabla u) + v \frac{\partial u}{\partial z} = f$$



$$f = 100$$

$$R = 1$$

$$v = v_0 \max\{0, 1 - (1-x)^5 - (1-y)^3\}$$

$$\left. \begin{array}{l} x=0 \\ y=0 \\ z=0 \end{array} \right\} \text{ディリクレ条件} \quad u=0$$

$$\left. \begin{array}{l} x=1 \\ y=1 \\ z=2 \end{array} \right\} \text{ノイマン条件} \quad \nabla u \cdot n = 0$$

図 3: テスト問題

表1から、セルペクレ数が大きくなって対角優位性が失われると BiCG 法に要する反復回数が増加すること、対称化した方程式に対する CG 法はセルペクレ数に対する依存性が弱いこと、BiCG 法の反復が増加するときに対称化した方程式に対する CG 法の反復回数は減少することがわかる。目安として、セルペクレ数が 8 のとき BiCG 法の 3 倍、セルペクレ数が 16 を越えると BiCG 法の 2 倍になる。

右辺の $b' = 0$ とおいた方程式 (5) に、異なる初期値 y_0 を与えた結果を以下に示す。

表 2: (5) に対する初期値 y_0 依存性 (反復回数)

	0(対称)	1	2	4	8	16	32	64
$(1, 0, \dots, 0)^T$	294	332		250		200		254
乱数	500	500	441	360	312	300	330	376

方程式系 (6) と (7) の比較を表 3 に示す。ここで、 $b' = b$, $y_0 = 0$ とした。

表 3: 方程式系 (6), (7) の収束 (反復回数)

	0(対称)	1	2	4	8	16	32	64
(6)	47	277	249	210	182	170	188	214
(7)	47	278	450	212	182	169	188	210

これらには、対称化したとき、非対称性の強い要素が対角要素の近くにくるかどうかという差があるが、いずれの場合にも非対角ブロック内にあるためか、系全体の収束には影響しなかった。

荻田の提案する方程式

$$\begin{pmatrix} A + A^T & A - A^T \\ -(A - A^T) & -(A + A^T) \end{pmatrix} \begin{pmatrix} x \\ x \end{pmatrix} = 2 \begin{pmatrix} b \\ -b \end{pmatrix}$$

に対して初期値を変えた結果を表 4 に示す。この場合、右辺に対する任意性はなく、アルゴリズムも図 1 と違ったものになる。

表 4: 荻田の方法に対する初期値 x_0 依存性 (反復回数)

	0(対称)	1	2	4	8	16	32	64
BiCG	46	52	47	54	71	89	116	140
$(1, 0, \dots, 0)^T$	259	331	295	246	204	202	242	260
乱数	500	500	448	358	319	300	352	374

4 まとめ

本研究では、3 次元移流拡散方程式を差分法で離散化して得られた非対称疎行列 A に対する方程式を対称化して CG 法を適用した。小さな $N = 250$ の問題に対する数値実験の結果から言えることは以下のとおりである。

- (1) 荻田の方法、 $b' = 0$ などの自由度の少ない方法の収束は芳しくない
- (2) BiCG 法の収束が芳しくない問題 (M 行列でないとき) には対称化すれば CG 法が使えるのである
- (3) 現時点では BiCG 法を上回る収束性能は得られていない
- (4) 小さな工夫で収束が変化するので、 b' や x_0 に対する解析が必要である

現時点での結論としては「試してみる価値がある」と言えよう。このアルゴリズムが有効かどうかの結論を下すには、

- 収束判定をどうすべきか
- 係数行列 A に対する依存性 (どのような問題に有効か)
- どの方程式系がよいのか? (どのように b' を構成するか)
- 初期値 x_0 , y_0 をどう選ぶか

などに対する検討が必要である。そのうえで、実用化には以下の点に対する調査が必要になってくるだろう。

- より現実的な大規模問題への適用
- 前処理の可能性: 全体への前処理 vs 個別の系への前処理
- 問題の正確な記述: 対称なら条件数が容易に得られる [11]
- 特化したプログラムによる実行性能の検討
- 桁数をふやすとどうなるか; 正規方程式を使ってはいけないのか?

参考文献

- [1] 曾我部知広, 鄭波, 橋本康, 張紹良, 非対称 Toeplitz 行列に対する CG 法の適用, 第 32 回数値解析シンポジウム講演予稿集, p. 59-62(2003)
- [2] 渡部善隆, 藤野清次, Persymmetric 行列に対する反復解法の収束性について, 日本応用数理学会 2003 年度年会講演予稿集, p. 138-139(2003)
- [3] 長谷川秀彦, クリロフ部分空間法の計算精度依存性, 日本応用数理学会 2003 年度年会講演予稿集, p. 316-317(2003)
- [4] E. H. Ayachour, Expanded systems and the ILU preconditioner for solving non-Hermitian linear systems, *Linear Algebra and its Applications*, pp. 243-256(1999)
- [5] M. Saunders, Solution of sparse rectangular systems using LSQR and CRAIG, *BIT*, 35:588-604, 1995.
- [6] M. Saunders, Solution of Sparse Linear Equations using Cholesky Factors of Augmented Systems, Report SOL 99-1, pp. 1-9, 1999.
- [7] 荻田武史, 大石進一, 対称拡大行列法の反復解法への適用, 日本応用数理学会 2003 年度年会講演予稿集, p. 318-319(2003)
- [8] 名取亮編, 長谷川秀彦, 櫻井鉄也, 桧山澄子, 周偉東, 花田孝郎, 北川高嗣, 数値計算法, オーム社, 1998, p. 16-20
- [9] 池田優介, 藤野清次, 柿原正伸, 井上明彦, A-直交過程に基づく RIF 前処理の効率化について, ハイパフォーマンスコンピューティングと計算科学シンポジウム HPCS2004, p. 1-8(2004)
- [10] R. Barrett, et al. 長谷川里美他訳, 反復法 Templates, 朝倉書店, 1996, p. 145-146
- [11] 長谷川秀彦, 対称正定値疎行列の条件数概算法, 日本応用数理学会論文誌, Vol. 1, No. 2, p. 169-176(1991)