2004－HPC－98　(1)
2004／4／13

接続問題から生ずる大規模な線形方程式の解法について：ランチョス法を中心にして

井田　献一朗[†]　野寺　隆[††]

本稿では偏微分方程式の境界値問題を離散化することで現れてくる行列方程式を解く方法について考える．使用するアルゴリズムは Single Seed Algorithm[5] と同じ形式をとるもので，ランチョス法と反復改良法を組み合わせたものである．行列方程式を解くには，行列方程式を多重の右辺をもった複数個の線形方程式とみなし，解くべき線形方程式（通常，これをシードシステムという）とそれ以外の線形方程式との 2 つに分けて考える．シード方程式はランチョス法により近似解を計算し，その他の方程式は反復改良法によって反復解を計算し，それをランチョス法での初期値として設定する手順を踏んで順に解を求める．このアルゴリズムは反復改良法で初期値を設定することでランチョス法の計算時間を減らすことができるという特徴を持っている．また，ランチョス法における近似解への収束を速めるために $\mathrm{ILL}^{\mathrm{T}}$ 分解による前処理を行う．数値実験では $\mathrm{ILL}^{\mathrm{T}}(0)$ と $\mathrm{ILL}^{\mathrm{T}}(1)$ の 2 通りの前処理を使用して 3 つの線形方程式からなる行列方程式を解き，提案手法の有効性を示す．

Method of solving large linear systems which occur in connection problems: Focusing on Lanczos method

Kenichiro IDA[†] and Takashi NODERA[††]

In this paper, we consider the solver of matrix equations which occur by discretizing baundary value problem of partial differential equations. The algorithm consists of the conbination of Lanczos method and iterative improvement, it is essentially based on *Single Seed Algorithm*[5]. A matrix equation is solved as follow that we regard it as linear systems with multiple right-hand side, which is divided into the equation to solve seed system, and others. Then seed system is computed by preconditioned Lanczos method and others are done by iterative improvement, which we set to initial guess of Lanczos method. The algorithm has the advantage that it can decrease computational time of Lanczos method by setting good initial guess using iterative improvement. The preconditioner of Lanczos method is used $\mathrm{ILL}^{\mathrm{T}}$ factorization. In our numerical experiments, we solve a matrix equation which consists of three linear systems using $\mathrm{ILL}^{\mathrm{T}}(0)$ or $\mathrm{ILL}^{\mathrm{T}}(1)$ preconditioner and show the effectiveness of the proposed algorithm.

1. 序　　論

自然現象の解明において現れる偏微分方程式は，解析解を求めることは一般に不可能であり，通常は近似解を求める手法をとることになる．近似解を求めるには偏微分方程式を有限差分法や有限要素法といった方法で離散化をすることになるが，それによって得られる線形方程式は大型で疎な係数行列を持ったものとなる．従って自然現象の解明には，この線形方程式を高速に解くアルゴリズムの開発が重要な役割を占める．線形方程式の解を求める方法には，直接法と反復法の 2 種類がある．直接法とは，ガウスの消去法のような有限回の操作で解が得られるものである．反復法は，真の解に収束するような近似解の列によって反復公式を構成し，その反復操作を何回か続けることで近似解を得る方法である．大型で疎な係数行列を持つ線形方程式の近似解を求める場合は，直接法よりも反復法を利用するほうが計算時間は大幅に速くなる．反復法にはさまざまなアルゴリズムが存在するが，対称正定値行列を係数とする連立 1 次方程式に対しては，CG 法やそれと数学的に同値なランチョス法が有効である．

これらの手法は線形な問題を解くために考えられたものであるが，自然現象の解析においては通常は非線形な問題が現れる．非線形な問題においては，線形方程式を複数個解かなければならないこともあるが，そのような問題を解く方法として，ブロック CG 法やブロックランチョス法などがある．これらのアルゴリズ

† 慶應義塾大学基礎理工学専攻
School of Fundamental Science and Technology, Keio University
†† 慶應義塾大学理工学部
Faculty of Science and Technology, Keio University

ムは複数個の方程式を同時に解く方法として利用されるが，問題によっては方程式が逐次的にしか得られない場合があり，そのような問題を解くのに適したアルゴリズムが必要になってくる．本稿ではそのアルゴリズムをランチョス法を通して考察する．

2. ランチョス法

2.1 ランチョス法のアルゴリズム

連立 1 次方程式

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

を考える．ただし，$A \in \boldsymbol{R}^{n\times n}$ は対称正定値で $x, b \in \boldsymbol{R}^n$ とする．さらに，x_0 を初期値とし，$r_0 = b - Ax_0, \beta_1 = \|r_0\|_2, v_1 = r_0/\beta_1$ とする．ここで，$\|\cdot\|_2$ はユークリッドノルムである．

ランチョス法とは次のような性質を持つクリロフ部分空間

$$\begin{aligned}\mathcal{K}_m(A, v_1) &= \mathrm{span}\{Av_1, \cdots, A^{m-1}v_1\} \\ &= \mathrm{span}\{v_1, \cdots, v_m\}\end{aligned}$$

を生成するアルゴリズムである．ここで，$v_1, \cdots, v_m$ は正規直交ベクトルでランチョスベクトルと呼ばれる．このランチョスベクトルから作られる行列 $V_m = [v_1, \cdots, v_m]$ を変換行列として

$$V_m^T A V_m = T_m$$

としたとき，T_m は対称な三重対角行列になる．このとき，(1) の近似解は次のように表すことができる．

$$x_m = x_0 + V_m T_m^{-1}(\beta_1 e_1)$$

ただし，e_1 は単位行列の最初の列のベクトルである．ランチョス法のアルゴリズムは次のようになる．

Algorithm 2.1 (ランチョス法)

(1) **set** $r_0 = b - Ax_0, \beta_1 = \|r_0\|_2,$
$v_0 = 0, v_1 = r_0/\beta_1$
(2) **for** $j = 1, \cdots, m$ **do**
(3) $v_{j+1} = Av_j - \beta_j v_{j-1}$
(4) $\alpha_j = v_j^T v_{j+1}$
(5) $v_{j+1} = v_{j+1} - \alpha_j v_j$
(6) $\beta_{j+1} = \|v_{j+1}\|_2$
(7) $v_{j+1} = v_{j+1}/\beta_{j+1}$
(8) **end for**
(9) **set** $V_m = [v_1, \cdots, v_m],$
$T_m = \mathrm{tridiag}(\beta_j, \alpha_j, \beta_{j+1})$
(10) **solve** $T_m y_m = \beta_1 e_1$
(11) **compute** $x_m = x_0 + V_m y_m$

ランチョス法で求まった近似解に対する残差ベクトルは，文献 1) より次のように計算できる．

$$r_m = b - Ax_m = \beta_{m+1} e_m^T y_m v_m$$

ただし，e_m は単位行列の m 列目のベクトルである．従って残差ノルムは $y_m^{(m)}$ を y_m の m 行目の成分とすると

$$\|r_m\|_2 = \beta_{m+1}|y_m^{(m)}|$$

と表すことができる．

2.2 行列の前処理

式 (1) に対して

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

$$AM^{-1}y = b \quad y = Mx \tag{3}$$

のように両辺に正則行列 M を掛けることを前処理という．前処理行列 M は，できるだけ A に近いものを選べば計算時間や反復回数を減らすことができる．また，M はできるだけ計算しやすいものを選ぶほうがよい．その条件を満たす方法の 1 つとして ILU 分解（不完全 LU 分解）がある．

2.2.1 ILU 分 解

行列 A を下三角行列 L と上三角行列 U の積

$$A = LU$$

に分解することを A の LU 分解という．LU 分解を行うとき，L, U のインデックスで A のゼロ成分のインデックスと同じところに非ゼロ成分が入ってくることがある．これをフィルインと言う．ILU 分解とは，このフィルインした成分をある程度無視して，それ以外の場所でのみ LU 分解を行うことである．すなわち，A の非ゼロ成分のインデックス集合

$$D = \{(i, j) | a_{ij} \neq 0\}$$

に対して，$S \supset D$ なる集合 S を 1 つ定め，S に含まれるインデックスに対してのみ LU 分解を行うことを言う．対称行列の ILU 分解は $\mathrm{ILL}^{\mathrm{T}}$ 分解という．特に $S = D$ としたときは $\mathrm{ILL}^{\mathrm{T}}(0)$ 分解と言う．フィルインした副対角を k 本残せば $\mathrm{ILL}^{\mathrm{T}}(k)$ 分解と呼ばれる．

2.2.2 前処理つきランチョス法

行列 A の $\mathrm{ILL}^{\mathrm{T}}$ 分解を前処理行列として前処理つきランチョス法を行うには，前処理は次のように行う．A の $\mathrm{ILL}^{\mathrm{T}}$ 分解を $M = LL^T$ としたとき，この M を分離させて

$$L^{-1}AL^{-T}y = L^{-1}b \quad y = L^T x$$

のように掛ける．これは係数行列の対称正定値性を保つためである．

Algorithm 2.2 (前処理つきランチョス法)

(1) **set** $\hat{r}_0 = L^{-1}r_0, \beta_1 = \|\hat{r}_0\|_2,$
$v_0 = 0, v_1 = \hat{r}_0/\beta_1$
(2) **for** $j = 1, \cdots, m$ **do**
(3) **solve** $L^T u_j = v_j$
(4) **solve** $L\hat{v}_j = Au_j$
(5) $v_{j+1} = \hat{v}_j - \beta_j v_{j-1}$
(6) $\alpha_j = v_j^T v_{j+1}$
(7) $v_{j+1} = v_{j+1} - \alpha_j v_j$
(8) $\beta_{j+1} = \|v_{j+1}\|_2$
(9) $v_{j+1} = v_{j+1}/\beta_{j+1}$

(10) **end for**
(11) **set** $V_m = [v_1, \cdots, v_m]$,
$T_m = \mathrm{tridiag}(\beta_j, \alpha_j, \beta_{j+1})$
(12) **solve** $T_m y_m = \beta_1 e_1$
(13) **solbe** $L^T z_m = V_m y_m$
(14) **compute** $x = x_0 + z_m$

3. 初期値設定の手法と行列方程式

3.1 行列方程式の解法

行列方程式

$$Ax^{(j)} = b^{(j)} \quad j = 1, \cdots, k$$

を考える．$A \in \boldsymbol{R}^{n\times n}$ は対称正定値で $x^{(j)}, b^{(j)} \in \boldsymbol{R}^n$ とする．この行列方程式を $j = 1$ から 1 つづつ近似解を求めたい．しかし，単純に解いていくだけでは計算量が 1 つの線形方程式を解くときの k 倍にもなってしまうので効率が悪い．そこで，1 つの方程式を解いている間，保留されている方程式に対しても何らかの処理を施しておけば計算量を減らせるかも知れない．あらかじめ保留されている方程式には, 解を求めるアルゴリズムで使用するための初期値を, ある程度解に近い値に設定する処理を施しておくことを考える．初期値が求めたい真の解に近ければ，計算時間も反復回数も減らすことができる．この考えに基づき，次のような方法を用いて行列方程式を解いてみる．

今，1 から $j-1$ 番目の方程式が解けたものとし，j 番目の方程式を A の $\mathrm{ILL^T}$ 分解を前処理行列に用いたランチョス法を使って解くものとする．ここで，この解くべき方程式のことをシードシステムと呼ぶ．その際，$j+1$ から k 番目の方程式に対しては，j 番目の方程式が解けるまで反復改良法と呼ばれるアルゴリズムを用いて，反復解を計算する．j 番目の方程式が解けたら，次に $j+1$ 番目の方程式をシードシステムとし，同じ手法を繰り返す．ただし $j+1$ 番目の方程式で用いる初期値は反復改良法で計算していた反復解を初期値として設定する．この操作を $j = 1$ から k まで行っていく．

3.2 反復改良法

反復改良法とは，$M = LL^T$ を A の $\mathrm{ILL^T}$ 分解としたとき，次の式で表される反復公式を用いて (1) の近似解を求めるアルゴリズムである．

$$x_n = x_{n-1} + M^{-1} r_{n-1} \tag{4}$$

この式を導く手順は次のとおりである．まず

$$B = I - M^{-1}A$$

とおく．このとき

$$A = M(I - B)$$

と書けるので $\|B\|_2 < 1$ であれば

$$\begin{aligned} A^{-1} &= (I-B)^{-1}M^{-1} \\ &= (I + B + B^2 + \cdots)M^{-1} \end{aligned} \tag{5}$$

となる．そこで

$$M_n^{-1} = (I + B + \cdots + B^n)M^{-1}$$

とおいて $x_n = M_n^{-1} b$ とすると

$$\begin{aligned} x_n &= B(I + B + \cdots + B^{n-1})M^{-1}b + M^{-1}b \\ &= Bx_{n-1} + M^{-1}b \\ &= (I - M^{-1}A)x_{n-1} + M^{-1}b \\ &= x_{n-1} + M^{-1}r_{n-1} \end{aligned}$$

となる．ただし，$r_{n-1} = b - Ax_{n-1}$ とおいた．アルゴリズムは次のようになる．

Algorithm 3.1 (反復改良法)
(1) **for** $k = 0, \cdots, m$ **do**
(2) **set** $r = b - Ax_k$
(3) **solve** $Ly = r$
(4) **solve** $L^T z = y$
(5) $x_{k+1} = x_k + z$
(6) **end for**

こうして求まった x_m をランチョス法の初期値として設定する．ただしこの反復解は $\|B\|_2 < 1$ を満たすときのみ収束するので A の性質によっては収束しないこともある．また前述の式 (5) を見ればわかるように, 収束の速さはノイマン級数並である．

3.3 アルゴリズム

ランチョス法と反復改良法を用いて 3.1 節で説明したようなアルゴリズムを構成すると Algorithm 3.2 のようになる．なお，ランチョス法のアルゴリズムは反復公式によって計算できる形式で実装している．その詳細は文献 1) を参照されたい．

4. 数 値 実 験

本章では楕円型偏微分方程式の境界値問題から得られる大型の行列方程式について述べる．

4.1 動 作 環 境

実験は次のような環境で行った．

OS Red Hat Linux 7.3
CPU Pentium 4 2.4GHz
メモリ 1GHz
計算精度 倍精度
前処理行列 $\mathrm{ILL^T}(0)$, $\mathrm{ILL^T}(1)$ の 2 通り
離散化時の刻み幅 $h = 1/200$
収束判定 $\|r_i^{(j)}\|_2 / \|r_0^{(j)}\|_2 < 10^{-12}$
ただし $r_i^{(j)}$ $(j = 1, \cdots, k)$ は j 番目の方程式がランチョス法で i 回反復したときの残差ベクトルを表す.

4.2 問 題

次のような楕円型偏微分方程式の境界値問題

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega = (0,1)^2 \\ u &= 0 \quad \text{on } \partial\Omega \end{aligned}$$

を考える．この方程式を 5 点中心差分によって離散化する．このとき，次のような係数行列が出てくる．

Algorithm 3.2

(1) **for** $l = 1, \cdots, k$ **do**
(2) **set** $\hat{r}_0^{(l)} = L^{-1} r_0^{(l)}$, $\zeta^{(l)} = \|\hat{r}_0^{(l)}\|_2$, $v_1^{(l)} = \hat{r}_0^{(l)}/\zeta^{(l)}$, $\lambda_1^{(l)} = \beta_1^{(l)} = 0, v_0^{(l)} p_0^{(l)} = 0$
(3) **for** $i = 1, 2, \cdots$ **until convergence do**
(4) **for** $j = l, l+1, \cdots, k$ **do**
(5) **if** $j = l$
(6) **solve** $L^T u_i^{(j)} = v_i^{(j)}$
(7) **solve** $L\hat{v}_i^{(j)} = A u_i^{(j)}$
(8) $v_{i+1}^{(j)} = \hat{v}_i^{(j)} - \beta_i^{(j)} v_{i-1}^{(j)}$
(9) $\alpha_i^{(j)} = {v_i^{(j)}}^T v_{i+1}^{(j)}$
(10) $v_{i+1}^{(j)} = v_{i+1}^{(j)} - \alpha_i^{(j)} v_i^{(j)}$
(11) $\beta_{i+1}^{(j)} = \|v_{i+1}^{(j)}\|_2$
(12) $v_{i+1}^{(j)} = v_{i+1}^{(j)}/\beta_{i+1}^{(j)}$
(13) **else**
(14) **set** $r^{(j)} = b^{(j)} - A x_0^{(j)}$
(15) **solve** $Ly = r^{(j)}$
(16) **solve** $L^T z = y$
(17) $x_0^{(j)} = x_0^{(j)} + z$
(18) **end if**
(19) **end do**
(20) **if** $i > 1$ **then compute** $\lambda_i^{(l)} = \beta_i^{(l)}/\eta_{i-1}^{(l)}$, $\zeta_i^{(l)} = \lambda_i^{(l)} \zeta_{i-1}^{(l)}$
(21) $\eta_i^{(l)} = \alpha_i^{(l)} - \lambda_i^{(l)} \beta_i^{(l)}$
(22) $p_i^{(l)} = (\hat{v}_i^{(l)} - \beta_i^{(l)} p_{i-1}^{(l)})/\eta_i^{(l)}$
(23) $x_i^{(l)} = x_{i-1}^{(l)} + \zeta_i^{(l)} p_i^{(l)}$
(24) **if** $\beta_{i+1}^{(l)} |\zeta_i^{(l)}| / |\eta_i^{(l)}| \|\hat{r}_0^{(l)}\|_2 < \varepsilon$ **then stop**
(25) **end do**
(26) **end do**

$$A = \begin{pmatrix} B & -I & & & \\ -I & B & -I & & \\ & \ddots & & & \\ & & -I & B & -I \\ & & & -I & B \end{pmatrix} \in \boldsymbol{R}^{39601 \times 39601}$$

ただし，I は単位行列で，行列 B は

$$B = \begin{pmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \ddots & & & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{pmatrix} \in \boldsymbol{R}^{199 \times 199}$$

のような行列である．この A を係数行列とした行列方程式 $Ax^{(j)} = b^{(j)}, j = 1, 2, 3$ を Algorithm 3.2 を使って解く．ただし，$b^{(j)} = [j, \cdots, j]^T$ とする．

4.3 実験結果

数値結果は各線形方程式の計算時間・反復回数・残差ノルムを**表 1** に記している．$\mathrm{ILL^T}(0)$ と $\mathrm{ILL^T}(1)$ それぞれの前処理つきランチョス法の収束の振る舞いを**図 1** と**図 2** にグラフにしている．表 1 をみると，初期値を反復改良法で設定したことで 2 つ目以降の方程式に対する計算時間は最初のそれと比べて半分以下になっている．しかし，図 1 と図 2 からわかるように，反復改良法の収束がノイマン級数並でそれほど早く収束しないため，ふたつ目以降の方程式の反復回数が最初のそれと比べてもあまり変わらないことである．従って，収束を速めるために式 (4) の前処理行列 M に加速パラメータをつけかえるなど，何らかの工夫が必要になってくる．ただし最適な加速パラメータを求めることはそれほど容易ではない．あるいは，反復改良法を適用する方程式の個数を制限してもよいかもし

表 1 Algorithm 3.2 による計算結果

Preconditioner	Linear system	Time (s)	Iterations	Residual norm
ILU(0)	1st	7.65	201	7.19×10^{-13}
	2nd	3.73	149	9.70×10^{-13}
	3rd	2.09	135	8.45×10^{-13}
ILU(1)	1st	4.78	136	9.61×10^{-13}
	2nd	2.30	95	8.47×10^{-13}
	3rd	1.21	83	9.41×10^{-13}

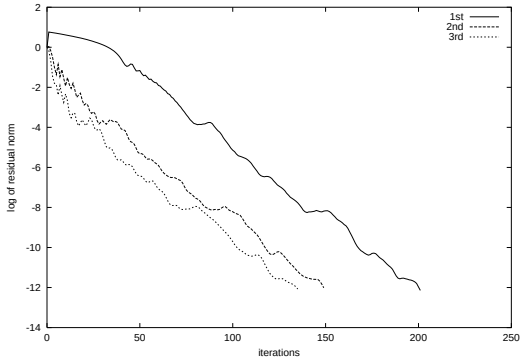

図 1 ILU(0) 分解を前処理行列に用いたランチョス法の収束の振る舞い

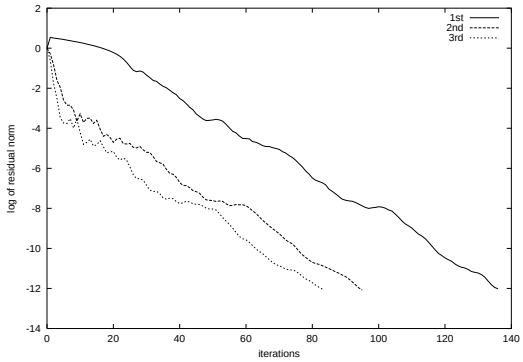

図 2 ILU(1) 分解を前処理行列に用いたランチョス法の収束の振る舞い

れない．たとえば,「3 つ目の方程式まで反復改良法を適用し，4 つ目以降の方程式は初期値を 0 と設定してランチョス法を適用する」などである．または，反復改良法自体を別のアルゴリズムと替えてみることも有効かもしれない．

5. おわりに

本稿では線形かつ対称な問題のみを取り扱ったが，ランチョス法は非対称な問題へ拡張することができるため一般の行列方程式に本稿の理論を適用することは可能である．また行列方程式を解くことは実際には非線形な偏微分方程式を離散化することで現れるものであり，決して線形な問題ばかりで利用されるものではない．ただ本稿で説明したアルゴリズムは反復改良法の収束に問題があり，アルゴリズムを広範囲な問題に応用するためには今後見直していくべき課題の 1 つとなる．オリジナルの Single Seed Algorithm では CG 法によりシードシステムを解き，ガレルキン射影法によって初期値を設定する，という手法をとっているので，なにも初期値の設定を反復改良法にこだわる必要はない．実際，反復改良法はノイマン級数並の収束しかしないという欠点があるので，もっと速い収束を示し，シードシステムの計算に負担がかからないように初期値の設定ができるアルゴリズムを構成することができればよりスマートに問題が解けるようになる．

参 考 文 献

1) Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company (1996).
2) L. Komzsik, *The Lanczos Method: Evaluation and Application*, SIAM (2003).
3) R. Barrett ほか著, 長谷川里美 ほか訳, 『反復法 Templates』, 朝倉書店 (1995).
4) R. S. Varga 著, 渋谷政昭 ほか訳, 『計算機による大型行列の反復解法』, サイエンス社 (1972).
5) C. S. Chien, S. L. Chang, *Application of The Lanczos Algorithm for Solving The Linear Systems that Occur in Continuation Problems*, Numerical Linear Algebla with Applications 2003; **10**: 335–355.
6) B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice–Hall, Inc. (1980).