ハイパフォーマンス
コンピューティング 66－7
（1997. 5. 9）

ブレイクダウンを起こさないランチョス法

森屋 健太郎† 野寺 隆†

ランチョス法は，大型疎行列の固有値を求めるための算法である．しかしながら，非対称行列を扱うときは，計算の途中でのブレイクダウン（算法の破綻）が，従来からの大きな問題点である．この論文では，この問題点を解決するための一つの方法として，再スタートベクトルを定義することによって，ブレイクダウンを回避する方法について述べる．この新しいランチョス法について，数値実験とともに報告する．

Breakdown-free Lanczos algorithm

KENTARO MORIYA † and TAKASHI NODERA†

Lanczos algorithm is a method to obtain some eigenvalues of large and sparse matrix. But in case of nonsymmetric matrix, breakdown has been a serious problem. In this paper, as one of the way to solve the problem, we avoid breakdown by defining a new start vector. The new algorithm and numerical examples are given.

1. はじめに

1950年にランチョスによって提案されたランチョス法は，大型疎行列の固有値問題に非常に有効な算法である．それは，次元が大きいにもかかわらず，計算の際に，記憶するべきベクトルの数が少なくて済むからである．しかも，大部分が行列とベクトルのかけ算，ベクトルの内積で構成させているので，並列計算機やベクトル計算機で計算させるのが非常に有効な算法である．その反面，数値的に不安定な算法であるため，ブレイクダウンが生じると，反復を進めることができなくなってしまうのが難点である．

しかし，再スタートベクトルを定義することにより，ブレイクダウンを回避できれば，この算法を続行することが可能である．以下このような新しい算法の改良について具体的に述べていくことにする．

2. 従来のランチョス法とその問題点

非対称行列の問題における，従来のランチョス法は，以下の式で示される．

$$\beta_{j+1}q_{j+1} = s_j = Aq_j - \alpha_j q_j - \gamma_j q_{j-1} \quad (1)$$

$$\gamma_{j+1}p_{j+1} = r_j = A^t p_j - \alpha_j p_j - \beta_j p_{j-1} \quad (2)$$

ただし，係数 α_j , β_j , γ_j は，$p_i^t q_j = \delta_{ij}$ （δ_{ij} はクロネッカーのデルタ）を満たすように定める．したがって，

$$\alpha_j = p_j^t A q_j$$
$$\beta_j = p_j^t A q_{j-1}$$
$$\gamma_j = p_{j-1}^t A q_j$$

である．次に，

$$P_m = (p_1, p_2, \ldots, p_m)$$
$$Q_m = (q_1, q_2, \ldots, q_m)$$

とすると，

$$P_m^t A Q_m = \begin{pmatrix} \alpha_1 & \gamma_1 & & \\ \beta_1 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \gamma_{m-1} \\ & & \beta_{m-1} & \alpha_m \end{pmatrix}$$

として，行列 A を三重対角化することができる．求まった三重対角行列を使って，行列 A の近似固有値が求められる．

ところで，ω_j のことを j 番目の反復におけるピボットとし，

$$\omega_j = \cos\langle s_j, r_j \rangle$$
$$= s_j^t r_j / (\| s_j \| \| r_j \|)$$

と定義する．ピボットの値 ω_j が，0 や極めて小さい値をとるときは，(1) , (2) 式で述べた，q_{j+1} か p_{j+1} の一方のベクトルが定義できなくなってしまう．これがランチョス法の従来からの問題点であるブレイクダウンであり，改善が必要な点である．

† 慶應義塾大学理工学部
Faculty of Science and Technology, University of Keio

3. 新しいランチョス法について

この節では，再スタートによって生じる新しい漸化式と，従来のランチョス法で生成された三重対角行列の代わりに生じるヘッセンベルグ行列のことについて述べる．

3.1 新しい再起式

再スタートベクトルを定義したとき，それは，最初に使った隣接する3項のベクトルによる漸化式で求められたものではない．したがって，もはやこの漸化式は無効である．そこで，新たに隣接4項による漸化式を定義して，この式を反復に用いる．これらの式は $j-1$ 番目でブレイクダウンが生じたとして，p_j を再スタートベクトルとすると，まず，j 番目の反復は以下の (3)，(4) 式でなされる．

$$\begin{aligned}\gamma_{j+1}^{(2)} p_{j+1} &= r_j \\ &= A^t p_{j-1} - \alpha_{j-1} p_{j-1} \\ &\qquad -\beta_{j-1} p_{j-2} - \gamma_j p_j \qquad (3)\\ \beta_{j+1} q_{j+1} &= s_j \\ &= A q_j - \alpha_j q_j - \gamma_j q_{j-1} \qquad (4)\end{aligned}$$

次に，$j+1$ 番目以降の反復は以下の (5)，(6) 式でなされる．ただし，$l \geq j+1$ である．

$$\begin{aligned}\gamma_{l+1}^{(2)} p_{l+1} &= r_l \\ &= A^t p_{l-1} - \alpha_{l-1} p_{l-1} \\ &\qquad -\beta_{l-1} p_{l-2} - \gamma_l p_l \qquad (5)\\ \beta_{l+1} q_{l+1} &= s_l \\ &= A q_l - \alpha_l q_l \\ &\qquad -\gamma_l q_{l-1} - \gamma_l^{(2)} q_{l-2} \qquad (6)\end{aligned}$$

また，$p_i^t q_j = \delta_{ij}$ なので，

$$\gamma_l^{(2)} = p_{l-2}^t A q_l$$

である．q_l の漸化式に関しては，j 番目の反復のときと $j+1$ 番目の反復のときとで異なっているが，これは，再スタートをした直後では，$\gamma_l^{(2)}$ の値がまだ定義されていないので，j 番目の反復では (6) 式が使えないからである．ゆえに，最初の j 番目の反復だけは (4) 式でする．そのため，ブレイクダウンを起こした直後では，p_l の漸化式と q_l の漸化式の項数は異なる．

次のブレイクダウンが生じるまで，(5)，(6) 式によって反復をする．そして，再びブレイクダウンが生じたときは，(5)，(6) 式に，新たに $\gamma_l^{(3)}$ を付け加えて，隣接5項の漸化式による反復を行なう．一般に，$k-1$ 回ブレイクダウンが生じるときは $\gamma_l^{(k)}$ が漸化式に付け加えられる．この算法は，一般にはアルゴリズム 3.1 のようになる．ただし，実際に再スタートさせることのできる回数は，たかだか数回程度である．なぜなら，再スタートする度に，記憶するべきベクトルの数が増えてしまい，不経済なためである．また，簡単のため，ベクトル q_l が常に単位ベクトルとなるように，β_l の値を定める．ϵ の値はブレイクダウンの基準となる値で，ピボット ω がこの値より小さくなったらブレイクダウンを起こしたとみなす．

[アルゴリズム 3.1]

(1) set $q_0 = 0$, $p_0 = 0$
(2) choose q_1 and p_1 as $p_1^t q_1 = 1$
(3) set $\acute{k} = 1$, $k = 1$, $\delta_1 = 1$
(4) **for** $l = 1$ **to** m
  (a) $\acute{k} = k$
  (b) $\alpha_l = p_l^t A q_l$
  (c) **for** $j = 1$ **to** $\acute{k}$
    (i) $\gamma_l^{(j)} = p_{l-j}^t A q_l$
    endfor
  (d) $s_l = A q_l - \alpha_l q_l - \sum_{j=1}^{\acute{k}} \gamma_l^{(\acute{k})} q_{l-j}$
  (e) $\beta_{l+1} = \| s_l \|$
  (f) $q_{l+1} = s_l / \beta_{l+1}$
  (g) $k = \acute{k} + \delta_l$
  (h) $r_l = A^t p_{l-k+1}$
    $-\alpha_{l-k+1} p_{l-k+1}$
    $-\beta_{l-k+1} p_{l-k}$
    $-\sum_{j=1}^{k-1} \gamma_{l-k+j+1}^{(j)} p_{l-k+j+1}$
  (i) **if** ($|r_l^t q_{l+1}| / \| r_l \| > \epsilon$) **then**
    (i) $p_{l+1} = r_l / r_l^t q_{l+1}$
    (ii) $\delta_l = 0$
  (j) **else**
    (i) choose restart vector p_{l+1}
    (ii) $\delta_l = 1$
    endif
  endfor

3.2 ヘッセンベルグ行列の生成

$\alpha_l = p_l^t A q_l \quad 1 \leq l \leq m$

を l 行 l 列の成分に，

$\beta_l = p_l^t A q_{l-1} \quad 2 \leq l \leq m$

を l 行 $l-1$ 列の成分に，

$$\gamma_l^{(i)} = p_{l-i}^t A q_l \quad 2 \leq l \leq m \quad 1 \leq i \leq l-1$$

を l 行 $l-1$ 列の成分にする行列を T_m とすると，行列 T_m はヘッセンベルグ行列となり，

$$\begin{aligned}A Q_m &= Q_m T_m + s_m e_{m,m}^t \qquad (7)\\ P_m^t A &= T_m P_m^t + e_{m-k+1,m} r_m^t \\ &\qquad + \ldots + e_{m,m} r_m^t \qquad (8)\end{aligned}$$

が成立する．ただし，

$P_m = (p_1, p_2, \ldots, p_m)$
$Q_m = (q_1, q_2, \ldots, q_m)$

であり，$\acute{k} = \acute{k}(m)$, $k = k(m)$ で，$e_{m,m}$ は，m 番目の要素が1である m 次の単位ベクトルである．(7) (8) 式はそれぞれ，アルゴリズム 3.1 の (4d) と (4h) を

$$Aq_l = s_l + \alpha_l q_l + \sum_{j=1}^{k} \gamma_l^{(k)} q_{l-j}$$

$$A^t p_{l-k+1} = r_l + \alpha_{l-k+1} p_{l-k+1} - \beta_{l-k+1} p_{l-k} - \sum_{j=1}^{k-1} \gamma_{l-k+j+1}^{(j)} p_{l-k+j+1}$$

と変形することから導くことができる.

さらに, $p_i^t q_j = \delta_{ij}$ より, 以下の二つの式が成り立つ.

$$P_m^t s(m,k) = Q_m^t r_m = Q_m^t r_m = \ldots = Q_m^t r_m = 0 \quad (9)$$

$$P_m^t Q_m = I_m \quad (10)$$

ただし, I_m は m 次の単位行列である.

(9) , (10) 式より, (7) 式の両辺の左側から行列 P_m^t をかけると,

$$P_m^t A Q_m = T_m \quad (11)$$

が成り立つ. もちろん, (8) 式の両辺の右側から行列 Q_m をかけることでも, (11) 式を導くことができる.

(11) 式は, ちょうど, 従来のランチョス法において生成される三重対角行列が, ヘッセンベルグ行列に代わっただけである. したがって, 従来のランチョス法では, (11) 式によって三重対角行列を生成するのに対して, 本稿の新しいランチョス法では, ヘッセンベルグ行列を生成する. また, 再スタートをするにしたがって, より多くの $\gamma_l^{(i)}$ の値が入りこんできて, T_m の上三角成分が密になっていく.

4. 再スタートの方法

前節で示したように, ブレイクダウンが生じたときには, 再スタートベクトルを定義して, ブレイクダウンを避けなければならない. その方法は, 大きく分けて二つある. それは, 再度直交化させるかさせないかである.

4.1 再直交化させて再スタートする方法

もともと, $p_i^t q_j = \delta_{ij}$ を満たしているので, $l-1$ 番目の反復でブレイクダウンが生じたときに定義される再スタートベクトル p_l も, $p_i^t q_j = \delta_{lj}$ を満たすように決めるのが一番よい方法である. 具体的な方法としては, グラムシュミットの直交化法などを使えばよい. しかし, このような方法は, 実際には非常に時間がかかり, あまり実用的でない. なぜなら, 前の反復で求まったベクトルをすべて記憶しなければならず, AP1000 のような分散メモリ型並列計算機で数値実験するには, 内積計算に対して, 非常に多くのプロセッサ間通信を必要としてしまうためである.

4.2 再直交化させないで再スタートする方法

再直交化させる算法は, 実際には役に立たない. そこで, 再直交化をさせないで再スタートさせる方法を考える. もともと, ランチョス法は反復をしていくにつれて, 丸め誤差の影響で双直交性を失ってしまう. また, 双直交性を失っても, アルゴリズム 3.1 の算法は有効である. l 回目の反復でブレイクダウンが生じたとき, 再スタートベクトル p_{l+1} を決める方法は,

$$p_{l+1}^t q_{l+1} = 1$$

のみを満たすように決定すればよい. 具体的にいうと, q_{l+1} はノルムが 1 のベクトルなので,

$$p_{l+1} = q_{l+1}$$

とおけばよい. この算法の優れている点は, 前に求めたベクトルを記憶しておく必要がないことと, 再スタートのときにプロセッサ間の通信をしなくて済むことである.

5. LR 分解と QR 分解について

ヘッセンベルグ行列を生成したら, その行列の固有値を求めることで, もとの行列の近似固有値が求まる. その方法には, LR 分解[1] と QR 分解[1] がある. LR 分解は, fill-in, つまり, もともと 0 である成分が 0 でなくなることはない算法であるが, 行列の対角成分に, どれかひとつでも 0 があると, 算法が破綻してしまう. それに対して, QR 分解は, fill-in が起こるが, 安定な算法である.

5.1 LR 分解

LR 分解は, ヘッセンベルグ行列 $T_m^{(i)}$ を対角成分が 1 である下二重対角行列 L_i と, 上三角行列 R_i に分解して,

$$T_m^{(i)} = L_i R_i$$

とおくと, 次の行列 $T_m^{(i+1)}$ を

$$T_m^{(i+1)} = R_i L_i$$

として反復する. この反復を繰り返すと, T_m の対角成分に固有値が出現する.

5.2 QR 分解

QR 分解は, LR 分解にでてくる L_i の代わりに, ユニタリー行列 Q_i を用いる. あとは基本的には同じである. fill-in が問題であるが, 鏡像変換[1] などで, これを少なくすることができる.

6. 数値実験

実験 1

実験 1 で扱う行列は,

$$A = \begin{pmatrix} B & 2B \\ 4B & 3B \end{pmatrix}$$

である. ただし,

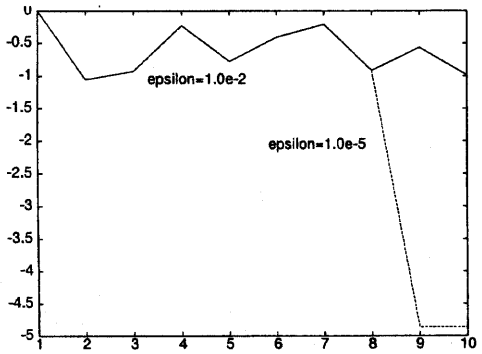

図 1　実験 1 におけるピボットの変化

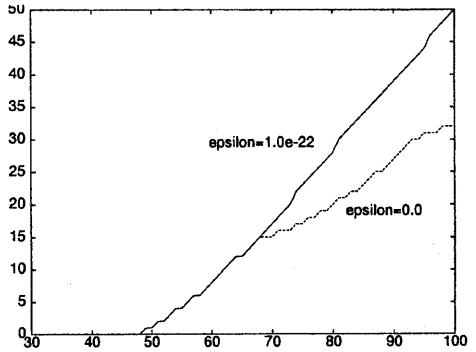

図 2　実験 2 における収束した固有値の数

表 1　実験 1 における固有値の誤差

	$\epsilon = 1.0 \times 10^{-2}$	$\epsilon = 1.0 \times 10^{-5}$
k	$\lvert\lambda_k - \theta_k\rvert$	$\lvert\lambda_k - \theta_k\rvert$
1	3.419043×10^{-12}	1.744983×10^{-9}
2	7.702494×10^{-8}	1.198127×10^{-3}
3	9.892679×10^{-8}	8.930390×10^{-4}
4	4.332166×10^{-8}	8.930040×10^{-4}
5	5.873631×10^{-8}	1.198108×10^{-3}
6	3.516458×10^{-8}	7.182149×10^{-4}
7	3.923979×10^{-7}	3.910410×10^{-7}
8	5.228997×10^{-7}	5.211367×10^{-7}
9	1.306901×10^{-7}	1.324556×10^{-7}
10	2.611919×10^{-7}	2.625575×10^{-7}

$$B = \begin{pmatrix} 0 & 1 & & & & \\ & 0 & 1 & & & \\ & & 0 & 1 & & \\ & & & 0 & 1 & \\ & & & & 0 & 1 \\ 10^{-5} & & & & & 0 \end{pmatrix}$$

とする．行列 A の固有値は，

$$\lambda_k = 0.5 \exp\{\, 2(k-1)\pi i \,/\, 5 \,\} \qquad k = 1, 2, \ldots, 5$$

$$\lambda_k = -0.1 \exp\{\, 2(k-6)\pi i \,/\, 5 \,\} \qquad k = 6, 7, \ldots, 10$$

である．

初期ベクトルは，

$$p_1 = q_1 = (\ 0,\ 1,\ 2,\ \ldots,\ 8,\ 9,\ )^t \,/\, \sqrt{285}$$

として，ϵ の値が 1.0×10^{-2} と 1.0×10^{-5} の場合について実験した．また，反復も 10 回完全に行なった．図 1 では，ピボット ω_l の変動を表すグラフを示している．縦軸が $\log|\omega_l|$ で，横軸が反復回数である．$\epsilon = 1.0 \times 10^{-5}$ の場合は，9 回目の反復で，ピボットが著しく減少している．これでは数値的に不安定である．

表 1 では，10 個の固有値それぞれの誤差を示している．明らかに $\epsilon = 1.0 \times 10^{-2}$ と $\epsilon = 1.0 \times 10^{-5}$ の場合では誤差の精度が異なる．これは，9 回目の反復で著しくピボット ω_l が減少しているにもかかわらず，再スタートをさせなかったためである．ピボット ω_l が小さい値になったり，減少が激しいときは数値的に不安定で，再スタートをせずにそのまま計算を続けると，良い精度の誤差が求まらないのである．しかし，実験 2 では必ずしも ϵ の値を大きくとればよいというわけではないことも分かった．

表 2　実験 2 における再スタートの回数

ϵ	1.0×10^{-12}	1.0×10^{-22}	0.0
回数	10	5	0

実験 2

領域 $\Omega = [0,1] \times [0,1]$ における楕円型偏微分方程式の境界値問題

$$-u_{xx} - u_{yy} + Du_x = f(x,y)$$
$$u(x,y)\ |_{\partial\Omega} = 1 + xy$$

を五点中心差分法で，離散化することによって得られる行列の固有値の収束性を調べる数値実験を行った．本実験では，分散メモリ型並列計算機 AP1000(使用したプロセッサは 32 個) を使用して，メッシュを 32×32 とし，$D = 64.0$ と設定した．

このようにして得られた行列に対し，本稿で述べた新しいランチョス法を適用した．初期スタートベクトル q_1 , p_1 は，第一成分だけが 1 である単位ベクトルとした．反復回数は最大 100 回として，30 回目以降から QR 分解によって近似固有値 θ_k を求めた．収束判定は，近似固有値の相対誤差で

$$|\theta_k - \theta_{k-1}| < 1.0 \times 10^{-12}$$

として行なった．ϵ の値は，1.0×10^{-12}, 1.0×10^{-22}, 0.0 の三種類の値を設定した．図 2 では，収束した固有値の数を示している．縦軸が収束した固有値の数で，横軸が反復回数である．なお，1.0×10^{-12} のときは値が大き過ぎて，たて続けに再スタートをしてしまい，ひとつも固有値が求まらなかった．表 2 では再スタートをした回数を表している．

7. 終 り に

再スタートをすれば, 一般にはよい精度の固有値が求まることが推測できるが, あまり大きな数値を ϵ の値に選んでしまっては, かえってブレイクダウンの回数が増えてしまい, 算法が破綻してしまう可能性がある. 前節の実験 2 で $\epsilon = 1.0 \times 10^{-12}$ とした例がそれである. この場合はたて続けに再スタートをしたために算法が破綻し, 固有値をひとつも求めることができなかった. 実際, この場合はブレイクダウンが起こっていないのである. どの値が適切かは, 問題にもよるので, いくつかの値を選んで試してみる必要がある.

参 考 文 献

1) 戸川 隼人, 「マトリクスの数値計算」オーム社. (1985).
2) B. N. Parlett, D. R. Taylor, and Z. A. Liu, *A look-ahead Lanczos algorithm for unsymmetric matrices,* Math. Comp. 44(1985), pp. 105-124.
3) J. H. Wilkinson, *The algebric eigenvalue problem,* Clarendon Press, Oxford, 1965.
4) J. Cullum, and R. A. Willoughby, *Lanczos algorithms for large symmetric eigenvalue computations,* Birkhäuser, Boston, 1985.
5) M. Gutknecht, *A completed theory of the unsymmetric Lanczos process and related algorithms, Part I,* SIAM J. Matrix Anal. Appl. 13 (1992), pp 594-639.
6) W. Kerner, *Large-scale complex eigenvalue problems,* J. Comput. Phys. 85 (1989), pp 1-85.
7) G. D. Smith, *Numerical Solution of Partial Differential Equations: Finite Difference Methods.* Oxford university Press, 1978.