数理解析研究所講究録
第1957巻 2015年 38-44

拡張テンソル和に対する最大・最小特異値計算 ～数値多重線形代数からのアプローチ～

大橋あすか　曽我部知広
愛知県立大学　大学院情報科学研究科
Asuka Ohashi and Tomohiro Sogabe
Graduate School of Information Science & Technology,
Aichi Prefectural University

1 はじめに

行列 $A \in \mathbb{C}^{\ell\times\ell}, B \in \mathbb{C}^{m\times m}$ のテンソル和は，テンソル積 "$\otimes$" と単位行列 I を用いて $I_m \otimes A + B \otimes I_\ell$ で定義され，その最大・最小特異値はシルベスター方程式（制御理論や2次元PDEの離散化等に現れる）の誤差解析に有用である．本研究が対象とする行列は，拡張されたテンソル和（以下，"拡張テンソル和"と略す）

$$T := I_n \otimes I_m \otimes A + I_n \otimes B \otimes I_\ell + C \otimes I_m \otimes I_\ell \in \mathbb{R}^{\ell mn \times \ell mn}$$

である．行列 T は大規模疎行列であり，行列 A, B, C が密行列で $\ell = m = n$ のとき，その非零成分の数は $O(n^4)$ で増加する．行列 T は3次元PDE：

$$[-\boldsymbol{a} \cdot (\nabla * \nabla) + \boldsymbol{b} \cdot \nabla + c]\, u(x,y,z) = g(x,y,z) \tag{1}$$

を7点中心差分法により離散化して得られる線形方程式の係数行列として現れる．ここで，$\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^3$, $c \in \mathbb{R}, \nabla * \nabla = [\partial^2/\partial x^2, \partial^2/\partial y^2, \partial^2/\partial z^2], u \in (0,1)\times(0,1)\times(0,1)$ とする．また，境界条件は全周Dirichlet条件である．

本研究では行列 T に対する最大・最小特異値の計算を目的とし，その効率化のために数値多重線形代数を用いた手法を考える．具体的には，基本的な反復法である（T や T^{-1} に対する）Lanczos 2重対角化法 [2] をテンソル空間上で考えることにより，所要メモリ量の削減に繋がる効率的な実装を行う．さらにテンソル空間上での初期値の提案により，効率的な実装を維持しつつ，収束の高速化が図られることを3次元PDEの離散化を例に検証する．

2 ベクトル空間 $\mathbb{C}^{\ell mn}$ 上のLanczos 2重対角化法

$M = T$ or T^{-1} に対するLanczos 2重対角化法のアルゴリズムの主要部を次に示す．

1: **for** $i = 0, 1, \ldots$ **do:**
2: $\quad \boldsymbol{r}_i := \underline{M^{\mathrm{H}}\boldsymbol{q}_i} - \alpha_i \boldsymbol{p}_i$; $\beta_i := \|\boldsymbol{r}_i\|_2$;
3: $\quad \boldsymbol{p}_{i+1} := \boldsymbol{r}_i/\beta_i$;
4: $\quad \boldsymbol{q}_{i+1} := \underline{M\boldsymbol{p}_{i+1}} - \beta_i \boldsymbol{q}_i$;
$\quad\quad \alpha_{i+1} := \underline{\|\boldsymbol{q}_{i+1}\|_2}$
5: $\quad \boldsymbol{q}_{i+1} := \boldsymbol{q}_{i+1}/\alpha_{i+1}$;
6: **end for**

ここで，行列 M^{H} は行列 M の共役転置を表す．

行列 T の最大・最小特異値の近似値は，このアルゴリズムで得られる α_i を対角に，β_i を副対角に持つ上二重対角行列の最大・最小特異値から得られる．また，一般にLanczos 2重対角化法の反復回数は，対象とする行列の行または列の数よりもはるかに少なくなる [1] ので，上二重対角行

列は行列 T よりも十分小さい行列となり，その特異値は特異値分解法などの直接法で計算可能である．

このアルゴリズムの所要メモリ量の主要部（下線部に対応）は，行列 A, B, C が密行列で $\ell = m = n$ と仮定すると，$M = T$ ならば行列 T の非零成分数から $O(n^4)$，$M = T^{-1}$ ならば直接法（例えば，LU 分解など）を用いると $O(n^6)$，反復法（例えば，Bi-CGSTAB 法，IDR(s) 法など）を用いると $O(n^4)$ となる．

3 3階のテンソルに対する演算

3階のテンソルとは，3次元配列のことであり，3階のテンソル $\mathcal{X}$ のサイズは $I \times J \times K$ と表される．以降では，“3階のテンソル”を“テンソル”と略す．

まずテンソルと行列との積であるモード積 [4] について説明する．$\mathcal{X} \in \mathbb{C}^{I \times J \times K}$ に対するモード積 “$\times_m$”（ただし，$m = 1, 2, 3$）は，それぞれ

$$(\mathcal{X} \times_1 U^{(1)})_{pjk} := \sum_{i=1}^{I} x_{ijk} u^{(1)}_{pi}, \quad (\mathcal{X} \times_2 U^{(2)})_{ipk} := \sum_{j=1}^{J} x_{ijk} u^{(2)}_{pj}, \quad (\mathcal{X} \times_3 U^{(3)})_{ijp} := \sum_{k=1}^{K} x_{ijk} u^{(3)}_{pk}$$

と定義される．ここで，$U^{(1)} \in \mathbb{C}^{P \times I}$, $U^{(2)} \in \mathbb{C}^{P \times J}$, $U^{(3)} \in \mathbb{C}^{P \times K}$, $i = 1, 2, \ldots, I$, $j = 1, 2, \ldots, J$, $k = 1, 2, \ldots, K$, $p = 1, 2, \ldots, P$ とする．次に，テンソルに対するノルムは，成分ごとの積であるアダマール積 “$*$” を用いて

$$\|\mathcal{X}\| := \sqrt{\sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{k=1}^{K} (\mathcal{X} * \mathcal{X})_{ijk}}$$

と定義される．最後に，ベクトルをテンソルに変換する逆 vec 作用素は

$$\mathrm{vec}^{-1} : \mathbb{C}^{\ell mn} \to \mathbb{C}^{\ell \times m \times n}$$

となる線形写像である．

4 テンソル $\mathbb{C}^{\ell \times m \times n}$ 上の Lanczos 2 重対角化法

本研究では，ベクトル空間 $\mathbb{C}^{\ell mn}$ 上の Lanczos 2 重対角化法をそれと同型なテンソル空間 $\mathbb{C}^{\ell \times m \times n}$ 上で考える．そこで，$M (= T \text{ or } T^{-1})$ に対する Lanczos 2 重対角化法に逆 vec 作用素を作用させると，Algorithm 1 が得られる．ここで，$\mathcal{P}_i = \mathrm{vec}^{-1}(\boldsymbol{p}_i), \mathcal{Q}_i = \mathrm{vec}^{-1}(\boldsymbol{q}_i), \mathcal{R}_i = \mathrm{vec}^{-1}(\boldsymbol{r}_i)$ であり，それぞれ $\ell \times m \times n$ のテンソルである．

4.1, 4.2 節において，Algorithm 1 における波線部の実装を説明する．

4.1 $M = T$ の場合

前節で示したモード積は，拡張テンソル和とベクトルの積の効率的な計算を可能にする．実際，モード積により波線部は

$$\begin{aligned} \mathrm{vec}^{-1}(T^{\mathrm{H}} \boldsymbol{q}_i) &= \mathcal{Q}_i \times_1 A^{\mathrm{H}} + \mathcal{Q}_i \times_2 B^{\mathrm{H}} + \mathcal{Q}_i \times_3 C^{\mathrm{H}}, \\ \mathrm{vec}^{-1}(T \boldsymbol{p}_i) &= \mathcal{P}_i \times_1 A + \mathcal{P}_i \times_2 B + \mathcal{P}_i \times_3 C \end{aligned} \tag{2}$$

となるため，小規模行列 A, B, C とテンソル $\mathcal{P}_i, \mathcal{Q}_i$ のみを用いて計算できる．結果として，波線部の所要メモリ量は主にテンソル $\mathcal{P}_i$, $\mathcal{Q}_i$ の所要メモリ量となり，$\ell = m = n$ と仮定すると $O(n^3)$ である．

Algorithm 1 テンソル空間上の Lanczos 2 重対角化法

1: $\mathcal{P}_0 \in \mathbb{C}^{\ell \times m \times n} (\|\mathcal{P}_0\| = 1)$ を選択する.
2: $\mathcal{Q}_0 := \mathrm{vec}^{-1}(M\boldsymbol{p}_0)$; $\alpha_0 := \|\mathcal{Q}_0\|$;
3: $\mathcal{Q}_0 := \mathcal{Q}_0/\alpha_0$;
4: **for** $i = 0, 1, \ldots$ **do:**
5: $\quad \mathcal{R}_i := \underbrace{\mathrm{vec}^{-1}(M^{\mathrm{H}}\boldsymbol{q}_i)} - \alpha_i \mathcal{P}_i$;
$\quad \beta_i := \|\mathcal{R}_i\|$;
6: $\quad \mathcal{P}_{i+1} := \mathcal{R}_i/\beta_i$;
7: $\quad \mathcal{Q}_{i+1} := \underbrace{\mathrm{vec}^{-1}(M\boldsymbol{p}_{i+1})} - \beta_i \mathcal{Q}_i$;
$\quad \alpha_{i+1} := \|\mathcal{Q}_{i+1}\|$;
8: $\quad \mathcal{Q}_{i+1} := \mathcal{Q}_{i+1}/\alpha_{i+1}$;
9: **end for**

4.2 $M = T^{-1}$ の場合

一般に，逆行列とベクトルとの積の計算には線形方程式を解く手法が用いられるが，本研究では逆行列を陽的に構成し，ベクトルとの積の計算をする手法を考える．

具体的な方法として，行列 T の右固有ベクトル・左固有ベクトルを列に持つ行列 X, Y，固有値を対角成分に持つ対角行列 Λ そしてブロック対角行列 $D = Y^{\mathrm{H}}X$ を用いる．このとき，逆行列 T^{-1} は

$$T^{-1} = X\Lambda^{-1}D^{-1}Y^{\mathrm{H}}$$

と分解可能である．この分解に用いられる行列は行列 T のテンソル構造を利用して求められるので，T^{-1} とベクトルとの積をテンソル空間上で実装する際に役立つ．以降では，各行列の求め方と実装について説明する．

まず行列 A に対する右・左固有ベクトルを列に持つ行列を $X_{(A)}, Y_{(A)}$ とし，行列 B, C についても同様に記述すると，行列 X, Y は行列 T の定義から，

$$X = X_{(C)} \otimes X_{(B)} \otimes X_{(A)}, \quad Y = Y_{(C)} \otimes Y_{(B)} \otimes Y_{(A)}$$

となる．次に，行列 A, B, C に対するブロック対角行列 $D_{(A)} := Y_{(A)}^{\mathrm{H}} X_{(A)}, D_{(B)} := Y_{(B)}^{\mathrm{H}} X_{(B)}$, $D_{(C)} := Y_{(C)}^{\mathrm{H}} X_{(C)}$ を用いると，逆行列 D^{-1} は

$$D^{-1} = D_{(C)}^{-1} \otimes D_{(B)}^{-1} \otimes D_{(A)}^{-1}$$

と計算される．すなわち，逆行列 D^{-1} は小規模なブロック対角行列の逆行列から得られる．最後に，行列 Λ が対角行列であることを利用して，$\mathcal{L} := \mathrm{vec}^{-1}(\mathrm{diag}(\Lambda^{-1}))$ とする．ここで，$\mathrm{diag}(\Lambda^{-1})$ は行列 Λ^{-1} の対角成分を順に並べて得られるベクトルを意味する．

以上より，波線部の実装は次のようになる．まず $\mathrm{vec}^{-1}((T^{-1})^{\mathrm{H}}\boldsymbol{q}_i)$ に対する実装は，

$$\begin{cases} \mathcal{Z}_1 = \mathcal{Q}_i \times_1 X_{(A)}^{\mathrm{H}} \times_2 X_{(B)}^{\mathrm{H}} \times_3 X_{(C)}^{\mathrm{H}}, \\ \mathcal{Z}_2 = \overline{\mathcal{L}} * \mathcal{Z}_1, \\ \mathcal{Z}_3 = \mathcal{Z}_2 \times_1 (D_{(A)}^{-1})^{\mathrm{H}} \times_2 (D_{(B)}^{-1})^{\mathrm{H}} \times_3 (D_{(C)}^{-1})^{\mathrm{H}}, \\ \mathrm{vec}^{-1}((T^{-1})^{\mathrm{H}}\boldsymbol{q}_i) = \mathcal{Z}_3 \times_1 Y_{(A)} \times_2 Y_{(B)} \times_3 Y_{(C)} \end{cases}$$

となり，次に $\mathrm{vec}^{-1}(T^{-1}\boldsymbol{p}_i)$ の実装は，

$$\begin{cases} \mathcal{Z}_4 = \mathcal{P}_i \times_1 Y_{(A)}^{\mathrm{H}} \times_2 Y_{(B)}^{\mathrm{H}} \times_3 Y_{(C)}^{\mathrm{H}}, \\ \mathcal{Z}_5 = \mathcal{Z}_4 \times_1 D_{(A)}^{-1} \times_2 D_{(B)}^{-1} \times_3 D_{(C)}^{-1}, \\ \mathcal{Z}_6 = \overline{\mathcal{L}} * \mathcal{Z}_5, \\ \mathrm{vec}^{-1}(T^{-1}\boldsymbol{p}_i) = \mathcal{Z}_6 \times_1 X_{(A)} \times_2 X_{(B)} \times_3 X_{(C)} \end{cases} \tag{3}$$

となる．ここで，$\overline{\mathcal{L}}$ はテンソル $\mathcal{L}$ の成分ごとの複素共役を表す．これらの実装により，波線部の所要メモリ量は $\ell = m = n$ のとき $O(n^3)$ となる．

4.3 初期値について

本小節では，Algorithm 1 に対する効率的な初期値について説明する．
本研究では，行列 A, B, C の右固有ベクトルの外積の凸結合で得られるテンソル

$$\mathcal{P}_0 = s_1 \left(\boldsymbol{x}_{i_{\mathrm{M}}}^{(A)} \circ \boldsymbol{x}_{j_{\mathrm{M}}}^{(B)} \circ \boldsymbol{x}_{k_{\mathrm{M}}}^{(C)} \right) + s_2 \left(\boldsymbol{x}_{i_{\mathrm{m}}}^{(A)} \circ \boldsymbol{x}_{j_{\mathrm{m}}}^{(B)} \circ \boldsymbol{x}_{k_{\mathrm{m}}}^{(C)} \right) \tag{4}$$

（ただし，$s_1, s_2 \geq 0$, $s_1 + s_2 = 1$）を初期値として提案する．ここで，$\{\lambda_i^{(A)}, \boldsymbol{x}_i^{(A)}\}, \{\lambda_j^{(B)}, \boldsymbol{x}_j^{(B)}\}, \{\lambda_k^{(C)}, \boldsymbol{x}_k^{(C)}\}$ を行列 A, B, C の固有値・右固有ベクトルとする．また，式 (4) において利用する右固有ベクトルの選び方を固有値を用いて，

$$i_{\mathrm{M}}, j_{\mathrm{M}}, k_{\mathrm{M}} = \arg\max_{i,j,k} \left\{ \left| \lambda_i^{(A)} + \lambda_j^{(B)} + \lambda_k^{(C)} \right| \right\},$$
$$i_{\mathrm{m}}, j_{\mathrm{m}}, k_{\mathrm{m}} = \arg\min_{i,j,k} \left\{ \left| \lambda_i^{(A)} + \lambda_j^{(B)} + \lambda_k^{(C)} \right| \right\}$$

（ただし，$i = 1, 2, \ldots, \ell$, $j = 1, 2, \ldots, m$, $k = 1, 2, \ldots, n$）とする．このような右固有ベクトルの選択によって，行列 A, B, C が対称行列のとき，初期値 $\mathcal{P}_0$ に vec 作用素を作用させて得られるベクトル $\boldsymbol{p}_0$ は行列 T の最大・最小特異値に対応する特異ベクトルの凸結合となる．

5 実装の評価と数値実験

5.1 実装の評価

本小節では，4.1, 4.2 節で示した式 (2), (3) とベクトル空間上での計算を所要メモリ量の観点で比較する．ベクトル空間上とテンソル空間上での実装について計算式と所要メモリ量を表 1 に示す．ただし，行列 A, B, C を密行列として得られる行列 T を対象とした．また，$M = T^{-1}$ に対するベクトル空間上での実装として，直接法と反復法を利用した場合の所要メモリ量を示した．

表 1: 所要メモリ量の比較（$\ell = m = n$）．

行列 M		ベクトル空間上	テンソル空間上
T	計算式	$T\boldsymbol{p}_i$	式 (2)
	所要メモリ量	$O(n^4)$	$O(n^3)$
T^{-1}	計算式	$T^{-1}\boldsymbol{p}_i$	式 (3)
	所要メモリ量	直接法：$O(n^6)$ 反復法：$O(n^4)$	$O(n^3)$

表1の所要メモリ量より，テンソル空間上の実装では所要メモリ量が$O(n^3)$であり，ベクトル空間上の実装に比べて省メモリ化が図られた．また，$M = T^{-1}$に対するテンソル空間上の実装では反復法を用いることなく省メモリな実装が可能となるので，計算量の観点で比較するとベクトル空間上と同程度かそれ以下となった．

5.2 数値実験

本小節では，4.3節で提案した式(4)で得られるテンソル（以下，“提案手法”と略す）と乱数を成分とするテンソル（以下，“乱数テンソル”と略す）を初期値とするAlgorithm 1の反復回数を比較し，提案手法による収束の高速化の検証を行う．なお，用いた実験環境はCPU: Intel Xeon X5650 2.67GHz，メモリ: 24GB，ソフト: MATLAB（R2011b）である．

数値実験で用いた行列は，式(1)を離散化して得られる行列のうち，対称性が高い行列と低い行列である．離散化で得られる行列の対称性は式(1)の$\boldsymbol{a}, \boldsymbol{b}, c$によって定まるので，対称性が高い行列として，$\boldsymbol{a} = [100, 100, 100]$, $\boldsymbol{b} = [1, 1, 1]$, $c = 1$と定めて得られた行列，対称性が低い行列として，$\boldsymbol{a} = [1, 1, 1]$, $\boldsymbol{b} = [100, 100, 100]$, $c = 1$と定めて得られた行列を利用した．これらの行列に対して，提案手法と乱数テンソルのいずれかを初期値とするAlgorithm 1に，式(2)または式(3)を用いて実験を行った．ここで，Algorithm 1の停止条件は残差ノルム：$\|M^{\mathrm{H}}\tilde{\boldsymbol{u}} - \tilde{\sigma}\tilde{\boldsymbol{v}}\|_2 = \beta_i |\boldsymbol{e}_i^{\mathrm{T}}\boldsymbol{u}| < 10^{-10}$（ただし，$\boldsymbol{e}_i$は$i$次元第$i$単位ベクトル，$\boldsymbol{u}$は上二重対角行列の特異値$\tilde{\sigma}$に対応する左特異ベクトル）を満たす場合とし，最大反復回数を4000回とした．

Algorithm 1が扱うテンソルサイズ（$n \times n \times n$）を$n = 5, 10, 15, 20, 25, 30$としたときの反復回数を表2から表5に示す．表2から表5の表記法として，nはAlgorithm 1が扱うテンソルサイズ（$n \times n \times n$），“×”は残差ノルムが条件を満たさず最大反復回数に達したこと，そして“*”は残差ノルムが条件を満たし停止したが，得られた特異値が最大または最小特異値でなかったことをそれぞれ意味する．得られた特異値が最大または最小特異値であるかを確かめるために，行列Tの最大・最小特異値を特異値分解法によって求め，Algorithm 1の結果との比較を行った．

表2: 対称性が高い行列における初期値による反復回数の変化（最大特異値）．

行列 M	初期値	n					
		5	10	15	20	25	30
T	乱数	27	56	81	109	132	162
	提案手法	16	29	41	100	64	145
T^{-1}	乱数	42	247	671	1473	2753	×
	提案手法	10	41	94	16	18	25

表3: 対称性が高い行列における初期値による反復回数の変化（最小特異値）．

行列 M	初期値	n					
		5	10	15	20	25	30
T	乱数	46	188	416	711	1063	1562
	提案手法	26	92	190	328	497	702
T^{-1}	乱数	6	6	6	6	6	6
	提案手法	3	4	4	4	4	4

まず表2, 3より，対称性が高い行列の最大・最小特異値計算に関して，提案手法を用いることで反復回数が削減された．特に最大特異値計算に着目すると，表2の$M = T^{-1}$を利用した結果のうち$n = 30$では，乱数テンソルを初期値とすると反復回数は4000回以上であったが，提案手法

表 4: 対称性が低い行列における初期値による反復回数の変化（最大特異値）.

行列 M	初期値	n					
		5	10	15	20	25	30
T	乱数	34	67	104	148	204	295
	提案手法	31	58	97	140	195	284
T^{-1}	乱数	67	246	513	1039	×	×
	提案手法	56	181	408	884	×	×

表 5: 対称性が低い行列における初期値による反復回数の変化（最小特異値）.

行列 M	初期値	n					
		5	10	15	20	25	30
T	乱数	97	206	293	382	487	620
	提案手法	95	217	321	410	523	653
T^{-1}	乱数	16	13	12	11	53*	445*
	提案手法	18	14	14	13	263*	334*

を用いることで 25 回に削減された．これらの結果から，対称性が高い行列の最大・最小特異値計算に対して，提案手法によって高速化されたことが確認された．

次に表 4, 5 より，対称性が低い行列の最大・最小特異値計算に関しては，どちらの初期値を利用しても同程度の反復回数であった．特に最小特異値計算では，提案手法を用いることにより反復回数は増加した．つまり，対称性が低い行列の最大・最小特異値計算に対しては，提案手法による高速化は得られなかった．

最後に表 5 から，対称性が低い行列の最小特異値計算について，$M = T^{-1}$ を利用した結果のうち $n = 25, 30$ に着目すると，残差ノルムが条件を満たして停止していたにも関わらず，得られた特異値は最小特異値ではないことが分かった．これは，式 (3) に行列の固有値・固有ベクトル分解を用いていることが原因の 1 つであると予想されるため，安定性の高い他の分解を用いた手法が必要であると考えられる．

6 まとめ

本研究では，拡張テンソル和 T の最大・最小特異値計算のために，テンソル空間上の Lanczos 2 重対角法を導出し，さらにテンソル空間上での初期値を提案した．

本研究により，大きく分けて 2 つの効果が得られた．第 1 の効果は省メモリ化である．行列 A, B, C を n 次正方行列かつ密行列とするとき，行列 T, T^{-1} に対して Lanczos 2 重対角化法を単純に適用するとその所要メモリ量は $O(n^4)$ となるが，これをテンソル空間上で実装することで，$O(n^3)$ に削減された．また，この実装の計算量は，単純に適用するときと同程度であった．第 2 の効果は高速化である．数値実験により，提案手法を初期値とするときテンソル空間上の Lanczos 2 重対角化法が高速化される場合があることが確認された．ただし対称性が低い行列 T に対する最大・最小特異値計算では，乱数を成分とするテンソルを初期値とするときと同程度もしくはそれより多くの反復回数を必要とした．

今後の課題としては，安定性の高い分解の 1 つである Schur 分解を利用した実装と対称性が低い行列に対する効率的な初期値の提案，そして本研究における手法をリスタート付き Lanczos 2 重対角化法 [1] や Jacobi-Davidson 型の特異値分解 [3] に応用することが挙げられる．

謝辞 本研究は 科学研究費補助金 基盤研究 (B) (課題番号 26286088) の補助を受けた．

参考文献

[1] J. Baglama and L. Reichel, Augmented implicitly restarted Lanczos bidiagonalization methods, SIAM J. Sci. Comput., 27 (2005), pp. 19-42.

[2] G.H. Golub and C.F. Van Loan, Matrix Computations, 3rd ed., JHU Press, 1996, pp. 495-496.

[3] M.E. Hochstenbach, A Jacobi-Davidson type SVD method, SIAM J. Sci. Comput., 23 (2001), pp. 606-628.

[4] T.G. Kolda and B.W. Bader, Tensor decompositions and applications, SIAM Rev., 51 (2009), pp. 455-500.