数理解析研究所講究録
第1976巻 2015年 81-90

行列Horner法の並列化による行列の固有ベクトル計算の効率化について

Improvement of efficiency of an algorithm for calculating eigenvectors of matrices with parallelized Horner's rule for matrices

田島 慎一*
SHINICHI TAJIMA
筑波大学 数理物質系
FACULTY OF PURE AND APPLIED SCIENCES, UNIVERSITY OF TSUKUBA

小原 功任†
KATSUYOSHI OHARA
金沢大学 理工研究域
FACULTY OF MATHEMATICS AND PHYSICS, KANAZAWA UNIVERSITY

照井 章‡
AKIRA TERUI
筑波大学 数理物質系
FACULTY OF PURE AND APPLIED SCIENCES, UNIVERSITY OF TSUKUBA

Abstract

Based on analysis of the residues of the resolvent, we have proposed an efficient algorithm for calculating eigenvector(s) of matrices. Our algorithm uses pseudo annihilating polynomials, and the elements in eigenvector are represented as a polynomial in eigenvalue as a variable, thus we do not need to find eigenvalues by solving the characteristic equation. We propose an improvement of efficiency of our algorithm in calculating a candidate of the eigenvector with parallelized Horner's rule for matrices.

1 はじめに

これまでに，我々は，レゾルベントの留数解析に基づき，行列の固有ベクトルを効率的に計算する算法を提案した ([3], [10]). 我々の算法は行列の最小消去多項式候補を用いるものであり，計算される固有ベクト

*tajima@math.tsukuba.ac.jp
†ohara@air.s.kanazawa-u.ac.jp
‡terui@math.tsukuba.ac.jp

ルの成分は，固有値を変数とする多項式で表され，しかもその次数は，固有値がみたす方程式の次数を超えることなく，取りうる最小値になるという特徴をもつ．最小消去多項式候補の算法は，著者（田島）らによるレゾルベントの留数解析に基づく効率的な算法が提案されている [4]．また，我々は，この固有ベクトル算法の実装において，“行列 Horner 法” の効率化 ([5], [6]) や，それらの並列実装 ([5], [7]) も提案している．その後，我々は，固有ベクトル算法のさらなる拡張を提案し，着目する固有値に属する一般固有ベクトル空間が，固有ベクトル空間に等しいという条件下で，着目する固有値の特性方程式における重複度が 1 よりも大きい場合に固有ベクトルを計算する算法の拡張を提案した ([8], [9])．

本稿では，我々が最初に提案した固有ベクトルの算法（着目している固有値の重複度が 1 に等しい場合に，その固有値に属する固有ベクトルを計算する算法）[10] において，固有ベクトル候補を計算する際の行列 Horner 法に並列処理を導入した，固有ベクトル計算のさらなる効率化を提案する．以下，第 2 章では，問題設定およびその解法について復習する．第 3 章では，固有ベクトル候補を計算する際の行列 Horner 法の並列化のアイデアを述べ，そのアルゴリズムを示す．第 4 章では，我々が提案するアルゴリズムを実装し，実験した結果を示す．

2 問題設定と固有ベクトルの算法

行列 A を有理数体 $\mathrm{K}=\mathbb{Q}$ 上の n 次正方行列とし，E_n を n 次単位行列とする．A の特性多項式 $\chi_A(\lambda)$ は次式の形で，K 上の既約因数分解があらかじめ求められているものとする:

$$\chi_A(\lambda) = f_1(\lambda)^{m_1} f_2(\lambda)^{m_2} \cdots f_p(\lambda)^{m_p} \cdots f_q(\lambda)^{m_q}. \tag{1}$$

本稿で用いるアルゴリズムの目的は，式 (1) のある既約因子 $f_p(\lambda)$ $(1 \le p \le q)$ に対し，$f_p(\alpha)=0$ をみたす A の固有値 $\lambda=\alpha$ に属する固有ベクトルを求めることである．なお，本稿では $m_p=1$ $(1 \le p \le q)$ とする．

n 次列ベクトル $\boldsymbol{u} \in \mathrm{K}^n$ に対し，$\{p(\lambda) \mid p(A)\boldsymbol{u}=\boldsymbol{0}\}$ をみたす多項式 $p(\lambda)$ は $\mathrm{K}[\lambda]$ のイデアルをなす．このとき，このイデアルの生成元をベクトル $\boldsymbol{u}$ に関する A の最小消去多項式と呼び，$\pi_{A,\boldsymbol{u}}(\lambda)$ で表す．$\boldsymbol{e}_j = {}^t(0,\ldots,0,1,0,\ldots,0)$ を，第 j 成分が 1 に等しい n 次単位ベクトルとし，列のインデックスを $J=\{1,2,\ldots,n\}$ とする．$\boldsymbol{e}_j$ に関する A の最小消去多項式 $\pi_{A,\boldsymbol{e}_j}(\lambda)$ は

$$\pi_{A,\boldsymbol{e}_j}(\lambda) = f_1(\lambda)^{l_{j,1}} f_2(\lambda)^{l_{j,2}} \cdots f_p(\lambda)^{l_{j,p}} \cdots f_q(\lambda)^{l_{j,q}}, \quad 0 \le l_{j,p} \le m_p, \quad j \in J \tag{2}$$

と表される．

本稿では，固有ベクトルの計算に，$\boldsymbol{e}_j$ に関する A の “最小消去多項式候補” $\pi'_{A,\boldsymbol{e}_j}(\lambda)$ を用いる．$\pi'_{A,\boldsymbol{e}_j}(\lambda)$ は

$$\pi'_{A,\boldsymbol{e}_j}(\lambda) = f_1(\lambda)^{l'_{j,1}} f_2(\lambda)^{l'_{j,2}} \cdots f_p(\lambda)^{l'_{j,p}} \cdots f_q(\lambda)^{l'_{j,q}} \tag{3}$$

と表される．ここに，我々の $\pi'_{A,\boldsymbol{e}_j}(\lambda)$ の求め方より，$\pi'_{A,\boldsymbol{e}_j}(\lambda)$ の各既約因子の多重度は $0 \le l'_{j,p} \le l_{j,p}$ を満たすことに注意する．

以下，$j \in J$ に対し，$\pi_{A,\boldsymbol{e}_j}(\lambda)$ を “A の第 j 列の最小消去多項式”，$\pi'_{A,\boldsymbol{e}_j}(\lambda)$ を “A の第 j 列の最小消去多項式候補” と呼ぶことにし，それぞれ $\pi_{A,j}(\lambda)$ および $\pi'_{A,j}(\lambda)$ で表す．また，$f_p(\lambda)$ を因子にもつような $\pi_{A,j}(\lambda)$, $\pi'_{A,j}(\lambda)$ に対し，それぞれ $g_{j,p}(\lambda)=\pi_{A,j}(\lambda)/f_p(\lambda)$, $g'_{j,p}(\lambda)=\pi'_{A,j}(\lambda)/f_p(\lambda)$ とおく．

$f_p(\lambda)$ に対し，2 変数多項式 $\Psi_p(x,y)$ を

$$\Psi_p(x,y) = \frac{f_p(x)-f_p(y)}{x-y}. \tag{4}$$

で定める．このとき，$\psi_p(x,y)$ は変数 y に関して $\deg(f_p)-1$ 次の多項式であることに注意する．

このとき，以下の命題が成り立つ．

命題 1

$\chi_A(\lambda)$, $f_p(\lambda)$, $\Psi_p(x,y)$, $\pi_{A,\boldsymbol{e}_j}(\lambda)$, $g_{j,p}(\lambda)$ を上記で与えられる多項式とする．このとき，列ベクトル $\rho(\lambda)$ を

$$\rho(\lambda) = \Psi_p(A, \lambda E) g_{j,p}(A) \boldsymbol{e}_j$$

によって定めると，$f_p(\alpha) = 0$ をみたす A の固有値 $\lambda = \alpha$ に対し，列ベクトル $\rho(\alpha)$ は

$$A \cdot \rho(\alpha) = \alpha \cdot \rho(\alpha), \tag{5}$$

をみたす．すなわち $\rho(\alpha)$ は A の固有値 $\lambda = \alpha$ に属する固有ベクトルである．∎

さて，本稿の算法では，最小消去多項式**候補**を用いて固有ベクトル計算を行う．よって，与えられた最小消去多項式候補が，実際に最小消去多項式であることを確認する必要がある．我々が提案した最小消去多項式候補を用いた固有ベクトル計算は，以下の流れになる．

アルゴリズム 1 (最小消去多項式候補を用いた固有ベクトル計算)

1. [**固有ベクトル候補の計算**] 注目している A の固有値を $\lambda = \alpha$, A の第 j 列の最小消去多項式候補で $f_p(\lambda)$ を因子にもつものを $\pi'_{A,j}(\lambda)$, $g'_{j,p}(\lambda) = \pi'_{A,j}(\lambda)/f_p(\lambda)$ とおく．このとき

   $$\rho(\lambda) = \Psi_p(A, \lambda E)\ g'_{j,p}(A)\ \boldsymbol{e}_j \tag{6}$$

   を計算する．

2. [**最小消去多項式のチェック**] $\pi'_{A,j}(\lambda)$ が A の第 j 列の最小消去多項式になるかどうかをチェックする．具体的には

   $$\pi'_{A,j}(A)\ \boldsymbol{e}_j = f_p(A)\ g'_{j,p}(A)\ \boldsymbol{e}_j \tag{7}$$

   が $\mathbf{0}$ に等しくなることを確かめる．

3. もし (7) が成り立つのであれば，(6) の $\rho(\lambda)$ を A の固有ベクトルとして出力する．

上記の手順で $\rho(\lambda)$ が 1 つ求まると，$f_p(\alpha) = 0$ をみたす任意の α に対し，$\rho(\alpha)$ は A の固有値 $\lambda = \alpha$ に属する固有ベクトルを表すことに注意する．

上記の手順の中で，固有ベクトル候補 $\rho(\lambda)$ を先に計算するのは，以下の理由による．(6) の $\Psi_p(A, \lambda E)$ から，Horner 法の 1 ステップの計算のみで，(7) の $f_p(A)$ が導かれる．この際，行列・ベクトル積の Horner 法を用いることで，固有ベクトル計算の効率化が可能になる（詳細は著者らによる先行論文 ([5], [6], [10]) を参照）．

固有ベクトル候補 $\rho(\lambda)$ の計算および最小消去多項式のチェックにかかる時間計算量（K 上の係数の演算回数のみを考慮する）は，行列 A の次元を n とするとき

$$O(n^2 \cdot \deg(\pi'_{A,j})) \tag{8}$$

となる [10].

3 固有ベクトル候補計算の並列化

本章では，アルゴリズム 1 のステップ 1 における式 (6) の $\rho(\lambda)$ の計算に着目する．$\Psi_p(A, \lambda E)$ は λ の $p-1$ 次式であるので，$\boldsymbol{u} = g'_{j,p}(A)\ \boldsymbol{e}_j$ とおき，式 (5) の固有値 α を λ に代入すると

$$\rho(\alpha) = \Psi_p(A, \alpha E)\boldsymbol{u} = \boldsymbol{u}_0\alpha^{p-1} + \boldsymbol{u}_1\alpha^{p-2} + \cdots + \boldsymbol{u}_{p-2}\alpha + \boldsymbol{u}_{p-1} \tag{9}$$

と表される．ここに，$\boldsymbol{u}_j$ $(j=0,\ldots,p-1)$ は，

$$f_p(\lambda)=a_0\lambda^p+a_1\lambda^{p-1}+\cdots+a_{p-1}\lambda+a_p \tag{10}$$

の各係数を用いて，次式で計算されるベクトルである．

$$\begin{aligned}
&\boldsymbol{u}_0=a_0\boldsymbol{u},\\
&\boldsymbol{u}_1=a_0A\boldsymbol{u}+a_1\boldsymbol{u} && =A\boldsymbol{u}_0+a_1\boldsymbol{u},\\
&\boldsymbol{u}_2=a_0A^2\boldsymbol{u}+a_1A\boldsymbol{u}+a_2\boldsymbol{u} && =A\boldsymbol{u}_1+a_2\boldsymbol{u},\\
&\quad\vdots && \quad\vdots\\
&\boldsymbol{u}_{p-1}=a_0A^{p-1}\boldsymbol{u}+a_1A^{p-2}\boldsymbol{u}+\cdots+a_{p-2}A\boldsymbol{u}+a_{p-1}\boldsymbol{u}=A\boldsymbol{u}_{p-2}+a_{p-1}\boldsymbol{u}.
\end{aligned} \tag{11}$$

式 (11) の各行の最右辺より，$\boldsymbol{u}_j$ は

$$\begin{cases}\boldsymbol{u}_0=a_0\boldsymbol{u},\\ \boldsymbol{u}_j=A\boldsymbol{u}_{j-1}+a_j\boldsymbol{u} \quad (j=1,\ldots,p-1)\end{cases}$$

のように，列ベクトルの左側に行列をかける形の Horner 法（これを “行列 Horner 法” と呼ぶ）によって計算される．この計算は，本来，j を 1 ずつ増加させながら逐次的に行うものであるが，本稿では，式 (11) の $\boldsymbol{u}_j$ の計算の順序を工夫することにより，計算の並列化が可能になることを示す．

まず，並列化可能な行列 Horner 法の計算手順を次の例題で示す．

例 1 ($\deg(f_p)=32$ **の場合の計算例)**

式 (10) において，$\deg(f_p)=32$ の場合における行列 Horner 法の計算例を示す．

1. 式 (11) のように求めるべきベクトル $\boldsymbol{u}_0,\ldots,\boldsymbol{u}_{31}$ を，以下に示す部分列に分解する．

$$L_1=[\boldsymbol{u}_0,\ldots,\boldsymbol{u}_7],\quad L_2=[\boldsymbol{u}_8,\ldots,\boldsymbol{u}_{15}],\quad L_3=[\boldsymbol{u}_{16},\ldots,\boldsymbol{u}_{23}],\quad L_4=[\boldsymbol{u}_{24},\ldots,\boldsymbol{u}_{31}]. \tag{12}$$

2. 式 (12) のリスト L_1 に属するベクトル $\boldsymbol{u}_0,\ldots,\boldsymbol{u}_7$ を，通常の（式 (11) と同様の）Horner 法を用いて求める．

$$\begin{aligned}
&\boldsymbol{u}_0=a_0\boldsymbol{u},\\
&\boldsymbol{u}_1=a_0A\boldsymbol{u}+a_1\boldsymbol{u} && =A\boldsymbol{u}_0+a_1\boldsymbol{u},\\
&\boldsymbol{u}_2=a_0A^2\boldsymbol{u}+a_1A\boldsymbol{u}+a_2\boldsymbol{u} && =A\boldsymbol{u}_1+a_2\boldsymbol{u},\\
&\quad\vdots && \quad\vdots\\
&\boldsymbol{u}_7=a_0A^7\boldsymbol{u}+a_1A^6\boldsymbol{u}+\cdots+a_6A\boldsymbol{u}+a_7\boldsymbol{u}=A\boldsymbol{u}_6+a_7\boldsymbol{u}.
\end{aligned} \tag{13}$$

3. 式 (12) のリスト L_2 に対応するベクトルの列 $L_2'=[\boldsymbol{u}_8',\ldots,\boldsymbol{u}_{15}']$ を，次式で求める．

$$\begin{aligned}
&\boldsymbol{u}_8'=a_8\boldsymbol{u},\\
&\boldsymbol{u}_9'=a_8A\boldsymbol{u}+a_9\boldsymbol{u} && =A\boldsymbol{u}_8'+a_1\boldsymbol{u},\\
&\boldsymbol{u}_{10}'=a_8A^2\boldsymbol{u}+a_9A\boldsymbol{u}+a_{10}\boldsymbol{u} && =A\boldsymbol{u}_9'+a_{10}\boldsymbol{u},\\
&\quad\vdots && \quad\vdots\\
&\boldsymbol{u}_{15}'=a_8A^7\boldsymbol{u}+a_9A^6\boldsymbol{u}+\cdots+a_{14}A\boldsymbol{u}+a_{15}\boldsymbol{u}=A\boldsymbol{u}_{14}'+a_{15}\boldsymbol{u}.
\end{aligned} \tag{14}$$

このとき，Horner 法の計算手順により，$j = 8, \ldots, 15$ に対し，

$$\boldsymbol{u}_j = A^{j-7}\boldsymbol{u}_7 + u'_j \tag{15}$$

が成り立つことに注意する．

4. 式 (12) のリスト L_3 に対応するベクトルの列 $L'_3 = [\boldsymbol{u}'_{16}, \ldots, \boldsymbol{u}'_{23}]$ を，次式で求める．

$$\begin{aligned}
&\boldsymbol{u}'_{16} = a_{16}\boldsymbol{u}, \\
&\boldsymbol{u}'_{17} = a_{16}A\boldsymbol{u} + a_{17}\boldsymbol{u} && = A\boldsymbol{u}'_{16} + a_{17}\boldsymbol{u}, \\
&\boldsymbol{u}'_{18} = a_{16}A^2\boldsymbol{u} + a_{17}A\boldsymbol{u} + a_{18}\boldsymbol{u} && = A\boldsymbol{u}'_{17} + a_{18}\boldsymbol{u}, \\
&\quad\vdots && \quad\vdots \\
&\boldsymbol{u}'_{23} = a_{16}A^7\boldsymbol{u} + a_{17}A^6\boldsymbol{u} + \cdots + a_{22}A\boldsymbol{u} + a_{23}\boldsymbol{u} = A\boldsymbol{u}'_{22} + a_{23}\boldsymbol{u}.
\end{aligned} \tag{16}$$

このとき，Horner 法の計算手順により，$j = 16, \ldots, 23$ に対し，

$$\boldsymbol{u}_j = A^{j-15}\boldsymbol{u}_{15} + u'_j \tag{17}$$

が成り立つことに注意する．

5. 式 (12) のリスト L_4 に対応するベクトルの列 $L'_4 = [\boldsymbol{u}'_{24}, \ldots, \boldsymbol{u}'_{31}]$ を，次式で求める．

$$\begin{aligned}
&\boldsymbol{u}'_{24} = a_{24}\boldsymbol{u}, \\
&\boldsymbol{u}'_{25} = a_{24}A\boldsymbol{u} + a_{25}\boldsymbol{u} && = A\boldsymbol{u}'_{24} + a_{25}\boldsymbol{u}, \\
&\boldsymbol{u}'_{26} = a_{24}A^2\boldsymbol{u} + a_{25}A\boldsymbol{u} + a_{26}\boldsymbol{u} && = A\boldsymbol{u}'_{25} + a_{26}\boldsymbol{u}, \\
&\quad\vdots && \quad\vdots \\
&\boldsymbol{u}'_{31} = a_{24}A^7\boldsymbol{u} + a_{25}A^6\boldsymbol{u} + \cdots + a_{30}A\boldsymbol{u} + a_{31}\boldsymbol{u} = A\boldsymbol{u}'_{30} + a_{31}\boldsymbol{u}.
\end{aligned} \tag{18}$$

このとき，Horner 法の計算手順により，$j = 24, \ldots, 31$ に対し，

$$\boldsymbol{u}_j = A^{j-23}\boldsymbol{u}_{23} + u'_j \tag{19}$$

が成り立つことに注意する．

6. 式 (14) および (15) を用いて，$L_2 = [\boldsymbol{u}_8, \ldots, \boldsymbol{u}_{15}]$ を求める．
7. 式 (16) および (17) を用いて，$L_3 = [\boldsymbol{u}_{16}, \ldots, \boldsymbol{u}_{23}]$ を求める．
8. 式 (18) および (19) を用いて，$L_4 = [\boldsymbol{u}_{24}, \ldots, \boldsymbol{u}_{31}]$ を求める．∎

例 1 の行列 Horner 法の計算は，以下の通り並列化可能である．

1. ステップ 2,...,5 における L'_2, L'_3, L'_4 は，互いに他のステップと独立して計算可能であるため，並列化可能である．
2. ステップ 6,...,8 における L_2, L_3, L_4 は，それぞれ $\boldsymbol{u}_7, \boldsymbol{u}_{15}, \boldsymbol{u}_{23}$ をあらかじめ計算しておくことにより，互いに他のステップと独立して計算可能であるため，並列化可能である．

以上より，固有ベクトル候補の計算における Horner 法の計算を並列化するアルゴリズムは次の通りまとめられる．

アルゴリズム 2 (行列-ベクトル Horner 法の並列計算による固有ベクトル候補の計算)
(注意：本アルゴリズムは，アルゴリズム 1, ステップ 1 より呼び出されるものである.)

入力
- A: n 次正方行列.
- $f_p(\lambda) = a_0\lambda^p + a_1\lambda^{p-1} + \cdots + a_{p-1}\lambda + a_p$: 行列 A の固有値のうち，計算しようとする固有ベクトルが属している固有値 α を零点にもつ特性多項式 $\chi_A(\lambda)$ の因子（式 (10) を参照）.
- $\boldsymbol{u} = g'_{j,p}(A)\, \boldsymbol{e}_j$: アルゴリズム 1 で求めておく固有ベクトル候補の "種"（式 (6) を参照）.

1. **[求めるベクトルの列を部分列に分割]** $L = [\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{p-1}]$ を s 個の部分列に分割する．それぞれの部分列に属するベクトルの個数を l_j $(j = 0, \ldots, s-1)$ とし，部分列への分割を

$$L_0 = [\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{l_0-1}], L_1 = [\boldsymbol{u}_{l_0}, \ldots, \boldsymbol{u}_{l_0+l_1-1}], \ldots, L_{s-1} = [\boldsymbol{u}_{l_0+\cdots+l_{s-2}}, \ldots, \boldsymbol{u}_{p-1}]. \tag{20}$$

とおく.

2. **[各部分列における途中結果のベクトルの計算]** 式 (20) における各部分列 $L_0, \ldots, L_{s-1}$ において，L_j に対する途中結果となるベクトルの列 $L'_j = [\boldsymbol{u}'_{l_0+\cdots+l_{j-1}}, \ldots, \boldsymbol{u}'_{l_0+\cdots+l_j-1}]$ を次式により計算する.

$$\begin{cases} \boldsymbol{u}'_{l_0+\cdots+l_{j-1}} = a_{l_0+\cdots+l_{j-1}}\boldsymbol{u}, \\ \boldsymbol{u}'_{l_0+\cdots+l_{j-1}+k} = A\boldsymbol{u}'_{l_0+\cdots+l_{j-1}+k-1} + a_{l_0+\cdots+l_{j-1}+k}\boldsymbol{u} \quad (k = 1, \ldots, l_j - 1). \end{cases}$$

なお，この時点で $L'_0 = L_0$ であり，L_0 の要素についてはさらなる計算は計算は不要であることに注意する.

3. **[各部分列の末尾の固有ベクトル候補の計算]** 上のステップで求めた部分列 L'_j $(j = 1, \ldots, s-1)$ に対し，L_j の末尾のベクトル $\boldsymbol{u}_{l_0+\cdots+l_j-1}$ を

$$\boldsymbol{u}_{l_0+\cdots+l_j-1} = A^{l_j}\boldsymbol{u}_{l_1+\cdots+l_{j-1}-1} + \boldsymbol{u}'_{l_0+\cdots+l_j-1}$$

により求める.

4. **[各部分列における最終結果のベクトルの計算]** $j = 1, \ldots, s-1$ に対し，部分列 L'_j における途中結果のベクトルから，部分列 L_j における最終結果となるベクトルを

$$\boldsymbol{u}_{l_1+\cdots+l_{j-1}+k} = A^{k+1}\boldsymbol{u}_{l_1+\cdots+l_{j-1}-1} + \boldsymbol{u}'_{l_1+\cdots+l_{j-1}+k} \quad (k = 0, \ldots, l_j - 2)$$

により求める.

5. **[固有ベクトル候補の出力]** 上のステップまでで計算した $\boldsymbol{u}_j$ $(j = 0, \ldots, p-1)$ を式 (9) に代入し，固有ベクトル候補 $\rho(\alpha)$ を出力する. ∎

3.1 並列化による計算時間の見積もりと効率化のための工夫

ここで，アルゴリズム 2 による時間計算量の見積もりを行い，計算の効率化のための工夫について考察する.

今回，行列 Horner 法における時間計算量の見積もりでは，行列-ベクトル積の計算量が全体の計算量を支配するので，アルゴリズムに現われる行列-ベクトル積を単位時間とした計算量を，逐次計算の場合と比較する.

例 2 ($\deg(f_p) = 256$ **の場合の時間計算量**)
式 (10) において $\deg(f_p) = 256$ の場合を取り上げる．逐次計算の行列 Horner 法の計算量は 255 単位時間である．

一方，並列処理数 8 で並列処理を行い，かつ，部分列の長さをすべて等しく分割すると，各部分列に属するベクトルの個数は 32 になる．よって，アルゴリズム 2 のステップ 2 および 4 において，各部分列に属するベクトルを並列計算するのにかかる計算量は，概算でそれぞれ $(256/8) \times 2 = 64$ 単位時間になる．

次に，ステップ 3 の部分は計算に依存性があるので，逐次計算を繰り返す必要がある．本計算例においては，A^8 をあらかじめ求めておけば，ステップ 3 は 7 単位時間で終了する．

以上により，本計算例の計算量は合計約 71 単位時間と見積もられる．∎

注意 1
部分列の個数が 2 で，かつ両方の部分列の長さが（ほぼ）等しい場合には，並列化した場合の計算量は逐次処理による計算量とほぼ等しくなると見積もられる．∎

また，ステップ 3 の計算の際に，部分列 L_j の個数が大きくなればなるほど，ステップ 4 の計算を待たされる部分列が発生する．そこで，部分列 L_j の中で，j が大きな（末尾に近い）部分列の長さを小さくし（他の部分列の半分程度），ステップ 3 の待ち時間を小さくしたり，空いている中央処理装置 (CPU) に処理を割り振ったりすることで，アルゴリズム全体の効率化を図ることが考えられる（実験結果も参照）．

4 実験

我々は，上述のアルゴリズム 2 を数式処理システム Risa/Asir に実装し，実験を行った．固有ベクトル計算に必要な諸計算のうち，特性多項式の計算には木村欣司氏による実装 [1] を，最小消去多項式候補の計算には著者による `taji_mat` パッケージ [5] を，Risa/Asir における並列計算には著者（小原）による `oh_p` パッケージ [2] をそれぞれ用いた．

実験環境は以下の通り: Intel Xeon E5-2690 at 2.90 GHz (8 cores) × 4, RAM 128GB, Linux 3.2.0 (SMP).

本稿の実験は，以下の要領で行った．

1. 与えられた行列の次元と特性多項式の因子の次数に対し，並列処理のコア数および式 (20) における部分列の分割の個数を変化させ，固有ベクトル計算中，並列処理による行列 Horner 法の部分の計算時間を計測し，それらを比較した．

2. 行列の次元は 128, 256, 512, 1024 とし，それぞれの次元において，行列の成分は十進 1 桁の整数の乱数で与えた．

3. 行列はその特性多項式が有理数上既約になるように与えたので，式 (10) は与えられた行列 A の特性多項式 $\chi_A(\lambda)$ に等しい．

4. アルゴリズム 2 におけるベクトルの部分列への分割は，すべての部分列の長さが等しくなるよう，均等に分割を行っている．

5. 計算時間の計測は，CPU 時間ではなく実時間を計測した（Risa/Asir の関数 `currenttime()`（実際には値を浮動小数で返す `dcurrenttime()`）を用いている）．計算時間の単位は秒である．

実験結果を表 1–4 に示す．表 1, 2, 3, 4 は，特性多項式の因子の次数がそれぞれ 128, 256, 512, 1024 の場合の実験結果を表す．表 1 および表 2 では，並列処理に用いた CPU のコア数は 1, 4, 8, 16 である一方，

表 3 および表 4 では，並列処理に用いた CPU のコア数として，上記の 1, 4, 8, 16 に加えて 32（実験環境における最大値）が加わっている．

各表において，行は並列処理に用いた CPU のコア数を表し，横軸は，式 (20) における部分列の分割の個数を表す．“1 コア” に該当する 1 行 1 列のデータは，従来の逐次計算による Horner 法の計算時間である（したがって部分列への分割は行っていない）．

実験結果から，以下のことがわかる．

1. どの実験においても，並列計算で部分列の分割数を 2 にした場合の計算時間が，逐次計算による Horner 法の計算時間とほぼ同一である．よって，注意 1 における指摘が実際の計算時間にも反映されていると見られる．

2. 部分列の分割数が 4 以上の場合において，並列化した行列 Horner 法の効果が現われている．特に，表 3 ($\deg(f_p) = 512$) および表 4 ($\deg(f_p) = 1024$) では，コア数が部分列の分割数よりも大きい場合において，計算時間が分割数にほぼ反比例しており，並列度の効果が高いと見られる．

3. いくつかの場合には，部分列の分割数が CPU のコア数を上回った場合に計算効率がより大きくなる（各表のデータで下線を引いた部分．表 2: コア数 8, 分割数 16; 表 3: コア数 4, 分割数 8, 16; コア数 8, 分割数 16, 32; コア数 16, 分割数 32; 表 4: コア数 4, 分割数 8, 16; コア数 8, 分割数 16, 32; コア数 16, 分割数 32）．第 3.1 節では，一部の部分列を分割することによるアルゴリズムの効率化を提案したが，この結果は，一部の部分列のみならず，部分列全体をより細かくすることで，計算効率がより向上する場合があることを示唆している．

5 まとめ

本稿では，レゾルベントの留数解析に基づいた行列の固有ベクトルの効率的算法において，行列 Horner 法を並列化することによる算法の効率化を提案した．Horner 法は本来逐次的な計算法であるが，我々の方法では，Horner 法の計算順序を工夫することで，逐次的な計算に依存しない部分を並列化可能であることを示した．計算機実験では，特に，特性多項式の因子の次数が 512 次や 1024 次といった高次の場合において，算法の並列化の効果がより大きなことを確かめた．

謝　辞

本稿の実験にあたり，行列の特性多項式計算プログラム [1] のソースコードをご提供下さった木村欣司氏に感謝いたします．

参 考 文 献

[1] K. Kimura. Linear Algebra PROGrams in Computer Algebra. http://www-is.amp.i.kyoto-u.ac.jp/kkimur/LAPROGCA.html (accessed November 4, 2015).

[2] 小原功任. OpenXM を用いた Risa/Asir 並列計算フレームワークの開発. 数式処理, **18**(1), 20–26, 2011.

[3] 田島慎一, 樋口水紀. レゾルベントを用いた固有ベクトル計算. Computer Algebra — Design of Algorithms, Implementations and Applications, 数理解析研究所講究録, 第 1666 巻, pp. 57–64. 京都大学数理解析研究所, 2009.

[4] 田島慎一, 奈良洸平. 最小消去多項式候補とその応用. Computer Algebra — Design of Algorithms, Implementations and Applications, 数理解析研究所講究録, 第 1815 巻, pp. 1–12. 京都大学数理解析研究所, 2012.

[5] Shinichi Tajima, Katsuyoshi Ohara, and Akira Terui. An extension and efficient calculation of the Horner's rule for matrices. In Proceedings of the 4th International Congress on Mathematical Software (ICMS 2014), Lecture Notes in Computer Science, 8592 pp. 346–351. Springer, 2014.

[6] 田島慎一, 小原功任, 照井章. 行列 Horner 法の拡張と効率化. 数式処理研究の新たな発展, 数理解析研究所講究録, 第 1930 巻, pp. 26–38. 京都大学数理解析研究所, 2015.

[7] 田島慎一, 小原功任, 照井章. 行列 Horner 法の並列化の実装について. 数式処理研究の新たな発展, 数理解析研究所講究録, 第 1930 巻, pp. 51–59. 京都大学数理解析研究所, 2015.

[8] 田島慎一, 照井章. 行列の最小消去多項式候補を利用した固有ベクトル計算 (II). 数式処理研究と産学研究の新たな発展, MI レクチャーノート, 第 49 巻, pp. 119–127. 九州大学マス・フォア・インダストリ研究所, 2013.

[9] 田島慎一, 照井章. 行列の最小消去多項式候補を利用した固有ベクトル計算 (III). 数式処理とその周辺分野の研究, 数理解析研究所講究録, 第 1907 巻, pp. 50–61. 京都大学数理解析研究所, 2014.

[10] 照井章, 田島慎一. 行列の最小消去多項式候補を利用した固有ベクトル計算. Computer Algebra — Design of Algorithms, Implementations and Applications, 数理解析研究所講究録, 第 1815 巻, pp. 13–22. 京都大学数理解析研究所, 2012.

# Cores	# of sublist(s)					
	1	2	4	8	16	32
1	0.53	—	—	—	—	—
4	—	0.65	0.39	0.37	0.40	—
8	—	0.64	0.47	0.30	0.35	0.83
16	—	0.74	0.51	0.32	0.36	0.86

表 1: $\dim(A) = 128$, $\deg(f_p) = 128$ の場合の計算結果（単位：秒）．詳細は第 4 章を参照.

# Cores	# of sublist(s)					
	1	2	4	8	16	32
1	7.95	—	—	—	—	
4	—	9.93	4.98	3.57	3.43	—
8	—	10.13	5.01	2.87	2.09	2.59
16	—	10.17	4.89	2.90	1.87	2.91

表 2: $\dim(A) = 256$, $\deg(f_p) = 256$ の場合の計算結果（単位：秒）．下線部は，部分列の分割数が CPU のコア数を上回った場合に計算がより効率化された部分を表す．詳細は第 4 章を参照.

# Cores	# of sublist(s)							
	1	2	4	8	16	32	64	128
1	163.47	—	—	—	—	—	—	—
2	—	205.92	91.60	87.56	—	—	—	—
4	—	199.49	104.61	51.10	48.36	—	—	—
8	—	196.23	102.76	53.11	29.14	26.12	—	—
16	—	193.08	101.95	50.75	25.30	20.45	26.55	—
32	—	190.47	100.82	53.72	26.68	20.36	27.34	60.36

表 3: $\dim(A) = 512$, $\deg(f_p) = 512$ の場合の計算結果（単位：秒）．下線部は，部分列の分割数が CPU のコア数を上回った場合に計算がより効率化された部分を表す．詳細は第 4 章を参照.

# Cores	# of sublist(s)							
	1	2	4	8	16	32	64	128
1	4033.23	–	–	–	–	–	–	–
2	–	4447.83	1348.02	1298.76	–	–	–	–
4	–	4176.92	2037.37	814.36	709.81	–	–	–
8	–	4094.72	2137.85	660.43	443.62	401.75	–	–
16	–	4251.86	2008.10	667.10	362.19	245.97	256.93	–
32	–	3885.93	2022.50	782.10	387.46	236.51	245.93	329.67

表 4: $\dim(A) = 1024$, $\deg(f_p) = 1024$ の場合の計算結果（単位：秒）．下線部は，部分列の分割数が CPU のコア数を上回った場合に計算がより効率化された部分を表す．詳細は第 4 章を参照.