Fast Algorithm for Calculating the Minimal Annihilating Polynomials of Matrices via Pseudo Annihilating Polynomials

Shinichi Tajima\textsuperscript{a}, Katsuyoshi Ohara\textsuperscript{b}, Akira Terui\textsuperscript{a,}\textsuperscript{*}

\textsuperscript{a}Faculty of Pure and Applied Sciences, University of Tsukuba, Tsukuba 305-8571, Japan
\textsuperscript{b}Faculty of Mathematics and Physics, Kanazawa University, Kanazawa 920-1192, Japan

Abstract

We propose an efficient method to calculate “the minimal annihilating polynomials” for all the unit vectors, of square matrix over the integers or the rational numbers. The minimal annihilating polynomials are useful for improvement of efficiency in wide variety of algorithms in exact linear algebra. We propose an efficient algorithm for calculating the minimal annihilating polynomials for all the unit vectors via pseudo annihilating polynomials with the key idea of binary splitting technique. Efficiency of the proposed algorithm is shown by arithmetic time complexity analysis.

Keywords: The minimal polynomial, The minimal annihilating polynomial, Exact calculation

2010 MSC: 15A18, 65F15, 68W30

\textsuperscript{*}Corresponding author

Email addresses: tajima@math.tsukuba.ac.jp (Shinichi Tajima),
ohara@se.kanazawa-u.ac.jp (Katsuyoshi Ohara), terui@math.tsukuba.ac.jp (Akira Terui)

URL: http://air.s.kanazawa-u.ac.jp/~ohara/ (Katsuyoshi Ohara),
http://researchmap.jp/aterui (Akira Terui)

February 6, 2018
1. Introduction

Linear algebra calculations over the integers and/or the rational numbers are important for various fields in mathematics, and a variety of software has been developed for the purpose ([1], [7], [10]).

We have proposed a series of algorithm based on residue calculus of resolvent of matrices for purposes such as calculating eigenvectors ([19], [22], [23], [24], [25]), (generalized) eigen decomposition ([3], [6], [8], [9], [13], [14], [15]), calculating matrix inverse ([20], [21]), spectral decomposition ([2], [3], [4], [5], [12]), and so on. We have shown that computational costs can be reduced significantly by using “the minimal annihilating polynomials” in place of the minimal polynomial ([16], [17]). Especially, it is very effective for spectral decomposition whose computational cost is high even using state-of-the-art technique of residue calculus.

While a matrix becomes zero by putting it into the variable in its characteristic or the minimal polynomial, such property preserves only for a specific column for the “unit” minimal annihilating polynomial, or the minimal annihilating polynomial for a unit vector. Since we need polynomials which makes only specific column(s) of the matrix to zero in the algorithms proposed so far, and such polynomials are factors of the characteristic or the minimal polynomial, using the minimal annihilating polynomials makes the algorithm efficient.

For designing efficient algorithms that utilizes the unit minimal annihilating polynomials, it is important to develop efficient algorithm for calculating all the unit minimal annihilating polynomials: this is what we deal with in this paper. Generally, direct calculation of the unit minimal annihilating polynomials is often time-consuming. Thus, in the proposed algorithm, we first calculate pseudo annihilating polynomials which are factors of true annihilating polynomials with almost deterministic method, then certify if it is true unit minimal annihilating polynomial; if the verification is not satisfied, then we can efficiently revise it to obtain true one. Furthermore, proposed algorithm has another benefit that certain processes are independent with each other so that these processes can be
executed *in parallel*, thus proposed algorithm fits into computing environments of multiple processors and/or cores to gain its efficiency.

The rest of the paper is organized as follows. In Section 2, we recall the (unit) minimal annihilating polynomial and give naive algorithm for calculating the one. In Section 3, we define pseudo annihilating polynomials which is factors of true minimal annihilating polynomials and give an algorithm for calculating the unit minimal annihilating polynomials via pseudo unit annihilating polynomials. In Section 4, we give efficient method in calculating pseudo unit annihilating polynomials using so-called binary splitting technique, then describe the main algorithm. Furthermore, we show that efficiency of the resulting algorithm is improved with the binary splitting technique by time complexity analysis of the algorithm.

2. The minimal annihilating polynomials

Let $\mathbb{K}$ be a field of characteristic zero, $A$ be a $n \times n$ matrix over $\mathbb{K}$ and $\mathbb{K}[\lambda]$ be a ring of univariate polynomials in $\lambda$ over $\mathbb{K}$. Let the irreducible factorizations of the characteristic polynomial of $A$ over $\mathbb{K}$ be

$$\chi_A(\lambda) = f_1(\lambda)^{m_1} f_2(\lambda)^{m_2} \cdots f_q(\lambda)^{m_q},$$

with $d_j = \deg(f_j)$ for $j = 1, \ldots, q$. Assume that we are already given the irreducible factorization of $\chi_A(\lambda)$.

We recall the minimal annihilating polynomial as follows.

**Definition 1** (The minimal annihilating polynomial). Let $A$ and $\mathbb{K}$ be the same as in the above, and $\mathbf{v} \neq \mathbf{0}$ be a column vector over $\mathbb{K}$ of dimension $n$. For an ideal $\text{Ann}_{\mathbb{K}[\lambda]}(A, \mathbf{v}) \subset \mathbb{K}[\lambda]$ defined as

$$\text{Ann}_{\mathbb{K}[\lambda]}(A, \mathbf{v}) = \{ p(\lambda) \in \mathbb{K}[\lambda] \mid p(A)\mathbf{v} = \mathbf{0} \},$$

we call the monic generator of $\text{Ann}_{\mathbb{K}[\lambda]}(A, \mathbf{v})$ the *minimal annihilating polynomial* of $A$ with respect to $\mathbf{v}$, denoted as $\pi_{A, \mathbf{v}}(\lambda)$. Furthermore, for $\mathbf{v} = \mathbf{e}_j$ where $\mathbf{e}_j$ is the $j$-th unit vector such that $\mathbf{e}_j = (0, \ldots, 0, 1, 0, \ldots, 0)$, we call $\pi_{A, \mathbf{e}_j}(\lambda)$ the $j$-th *unit* minimal annihilating polynomial, denoted as $\pi_{A,j}(\lambda)$. 

3
Let the factorization of the \( j \)-th unit minimal annihilating polynomial \( \pi_{A,j}(\lambda) \) be
\[
\pi_{A,j}(\lambda) = f_1(\lambda)^{r_{j,1}} f_2(\lambda)^{r_{j,2}} \cdots f_q(\lambda)^{r_{j,q}},
\]
(3)
\[0 \leq r_{j,i} \leq m_i \text{ for } i = 1, \ldots, q.\]

For \( p \) satisfying \( 1 \leq p \leq q \), let
\[g_p(\lambda) = f_1(\lambda)^{m_1} f_2(\lambda)^{m_2} \cdots f_{p-1}(\lambda)^{m_{p-1}} f_{p+1}(\lambda)^{m_{p+1}} \cdots f_q(\lambda)^{m_q}
= \chi_A(\lambda)/(f_p(\lambda)^{m_p}),\]
(4)
\[G_p = g_p(A), \quad F_p = f_p(A).\]

Then, in the \( j \)-th unit minimal annihilating polynomial in eq. (3), the exponent \( r_{j,p} \) of the factor \( f_p(\lambda) \) is identified as the minimum \( k \) satisfying \( F_p^k G_p e_j = 0 \).

With this property, a naive algorithm for calculating the unit minimal annihilating polynomial(s) is given as in Algorithm 1.

**Remark 1.** If we already know the minimal polynomial of \( A \) along with its irreducible factorization as
\[
\pi_A(\lambda) = f_1(\lambda)^{l_1} f_2(\lambda)^{l_2} \cdots f_q(\lambda)^{l_q},
\]
(5)
then \( m_i \) in eq. (4) are replaced with \( l_i \). (Efficient algorithms for calculating the minimal polynomial (e.g. [11]) have been proposed.)

In this paper, time complexity of algorithms is estimated with arithmetic operations in \( \mathbb{K} \), assuming that the irreducible factorization of \( \chi_A(\lambda) \) is given unless otherwise stated.

Note that the Horner’s rule can be used for evaluating a univariate polynomial by a matrix followed by multiplying a column vector from the right efficiently, as follows.

**Proposition 1.** Let \( f(x) \in \mathbb{K}[x] \) be
\[
f(x) = a_d x^d + a_{d-1} x^{d-1} + \cdots + a_0 x^0,
\]
(6)
with \( a_d \neq 0 \), \( A \in \mathbb{K}^{n \times n} \) and \( \mathbf{v} \in \mathbb{K}^n \) be a column vector. Then, a vector \( f(A)\mathbf{v} \) is calculated in \( O(d^2 m) \) arithmetic operations in \( \mathbb{K} \).
Algorithm 1 Calculating the $j$-th unit minimal annihilating polynomial $\pi_{A,j}(\lambda)$

\textbf{Input:} $A \in \mathbb{K}^{n \times n}$; \quad \triangleright \text{Input matrix;}
\chi_A(\lambda) = f_1(\lambda)^{m_1} f_2(\lambda)^{m_2} \cdots f_q(\lambda)^{m_q} \in \mathbb{K}[\lambda]; \quad \triangleright \text{Irreducible factorization of the characteristic polynomial of } A \text{ as in eq. (1)};

\textbf{Output:} \{r_{j,1}, \ldots, r_{j,q}\}; \quad \triangleright \text{The list of exponents of the factors in } \pi_{A,j}(\lambda) \text{ as in eq. (3)}

1: \textbf{for} $i = 1, \ldots, q$ \textbf{do}
2: \quad \quad \quad g_i(\lambda) \leftarrow \chi_A(\lambda)/(f_i(\lambda)^{m_i});
3: \quad \quad \quad b_{i,j} \leftarrow \text{Matrix-vector-horner}(g_i(\lambda), A, e_j); \quad \triangleright \text{ } b_{i,j} \leftarrow g_i(A)e_j
4: \quad \quad \quad k \leftarrow 0;
5: \quad \quad \textbf{while } b_{i,j} \neq 0 \textbf{ do}
6: \quad \quad \quad b_{i,j} \leftarrow \text{Matrix-vector-horner}(f_i(\lambda), A, b_{i,j}); \quad \triangleright \text{ } b_{i,j} \leftarrow f_i(A)b_{i,j}
7: \quad \quad \quad k \leftarrow k + 1;
8: \quad \quad \textbf{end while}
9: \quad \quad r_{j,i} \leftarrow k;
10: \textbf{end for}
11: \textbf{return} \{r_{j,1}, \ldots, r_{j,q}\}.

Proof. $f(A)v$ is calculated with the Horner’s rule with incorporating multiplication of $v$ from the right as

$$f(A)v = (a_d A^d + a_{d-1} A^{d-1} + \cdots + a_0 E)v$$

$$= A(\cdots A(A(a_n Av + a_{d-1} v) + a_{d-2} v) \cdots) + a_0 v,$$

with repeating pairs of a matrix-vector multiplication and a vector addition, whose complexity is $O(d^2)$ and $O(d)$, respectively, for $m$ times, whose const is bounded by $O(d^2m)$ in total.

Corollary 2. Let $f(x)$ be the same as in eq. (6), and $w \in \mathbb{K}^n$ be a row vector. Then, a vector $w f(A)$ is calculated in $O(d^2m)$ arithmetic operations in $\mathbb{K}$. 

5
Proof. Calculation in eq. (7) is rearranged as

\[ w f(A) = w(a_d A^d + a_{d-1} A^{d-1} + \cdots + a_0 E) \]
\[ = (\cdots ((a_d (w A) + a_{d-1} w) A + a_{d-2} w) A \cdots ) A + a_0 w, \]

thus the amount of arithmetic operations is the same as that in Proposition 1.

We summarize Proposition 1 and Corollary 2 as in Algorithms 2 and 3, respectively, for use in other algorithms in this paper.

**Algorithm 2** The Horner’s rule for matrix polynomial multiplied by a column vector from the right side

**Input:** \( f(x) = a_m x^m + a_{m-1} x^{m-1} + \cdots + a_0 x^0 \in \mathbb{K}[x]; \)
\( A \in \mathbb{K}^{n \times n}; \)
\( v \in \mathbb{K}^n; \) \( \leadsto \) A column vector

**Output:** \( f(A)v; \)

1: **function** Matrix\_vector\_horner\((f(x), A, v)\)
2: \( \text{return } f(A)v \) calculated as in eq. (7).
3: **end function**

**Algorithm 3** The Horner’s rule for matrix polynomial multiplied by a row vector from the left side

**Input:** \( f(x) = a_m x^m + a_{m-1} x^{m-1} + \cdots + a_0 x^0 \in \mathbb{K}[x]; \)
\( A \in \mathbb{K}^{n \times n}; \)
\( w \in \mathbb{K}^n; \) \( \leadsto \) A row vector

**Output:** \( wf(A); \)

1: **function** Vector\_matrix\_horner\((f(x), A, w)\)
2: \( \text{return } wf(A) \) calculated as in eq. (8).
3: **end function**

**Remark 2.** We have proposed “extended Horner’s rule” [18] for efficient calculation of Horner’s rule for matrix polynomials and vectors by reducing the
number of matrix-matrix multiplications with precalculation of certain powers of matrix.

**Proposition 3.** For given matrix $A \in \mathbb{K}^{n \times n}$ and irreducible factorization of its characteristic polynomial $\chi_A(\lambda)$, the $j$-th minimal annihilating polynomial $\pi_{A,j}(\lambda)$ is calculated by Algorithm 1 with

$$O((q - 1)n^3 + n^2 \deg(\pi_{A,j}(\lambda)))$$

arithmetic operations in $\mathbb{K}$.

**Proof.** We first estimate time complexity for calculating $b_{i,j}$ in lines 2 and 3, which can be calculated as

$$b_{i,j} = g_i(A)e_j = \prod_{k=1, k \neq i}^q f_k(A)^{m_k} e_j$$

$$= (f_1(A)^{m_1}) \cdots (f_{i-1}(A)^{m_{i-1}})(f_{i+1}(A)^{m_{i+1}}) \times \cdots$$

$$\cdots \times (f_{q-1}(A)^{m_{q-1}})(f_q(A)^{m_q} e_j),$$

by repeating the Horner’s rule with multiplying a vector as in Proposition 1 for $k = q, q-1, \ldots, i+1, i-1, \ldots, 1$ which costs $O(n^2(n - d_i m_i))$ arithmetic operations. Repeating this calculation for $i = 1, \ldots, q$ in the “for” loop from line 1 requires

$$\sum_{i=1}^{q} O(n^2(n - d_i m_i)) = O(n^2(qn - \sum_{i=1}^{q} d_i m_i)) = O(n^2(q - 1)n)$$

$$= O((q - 1)n^3).$$

Next, in line 6 calculating $b_{i,j} = f_i(A) b_{i,j}$ requires $O(n^2 d_i)$ operations, and by repeating this calculation for $r_{j,i}$ times in the “while” loop, we require $O(r_{j,i} n^2 d_i)$ operations in total. Repeating these calculations for $i = 1, \ldots, q$ in the “for” loop from line 1 requires

$$\sum_{i=1}^{q} O(r_{j,i} n^2 d_i) = O\left(n^2 \sum_{i=1}^{q} (r_{j,i} d_i)\right) = O(n^2 \deg(\pi_{A,j}(\lambda))).$$

Sum of the amounts in eqs. 11 and 12 gives an estimate of the number of operations in the whole algorithm as in eq. 9, which proves the proposition. □
Time complexity of calculating all of the minimal annihilating polynomials of $A$ with Algorithm 1 is equal to

$$O \left( (q - 1)n^4 + n^2 \sum_{j=1}^{n} \deg(\pi_{A,j}(\lambda)) \right)$$  \hspace{1cm} (13)

which is the sum of estimates in eq. (9) for $j = 1, \ldots, n$. In eq. (13), especially the first term is time-consuming for calculating $b_{i,j}$ as in eq. (10). To overcome this issue, we introduce \textit{pseudo} annihilating polynomials in the next section.

3. Pseudo annihilating polynomials for calculating minimal annihilating polynomials

Pseudo annihilating polynomial is defined as follows.

\textbf{Definition 2 (Pseudo annihilating polynomial).} Let $A$, $v$ and $K$ be the same as in the above, and let $u$ be a row vector of dimension $n$ over $K$ and $p(\lambda) \in K[\lambda]$. If $u$, $v$ and $p(\lambda)$ satisfy

$$up(A)v = 0, \quad u \neq 0, \quad v \neq 0, \quad p(\lambda) \mid \pi_{A,v}(\lambda),$$

then we call $p(\lambda)$ pseudo annihilating polynomial of $A$ with respect to $u$ and $v$, denoted as $\pi'_{A,v,u}(\lambda)$. Furthermore, for $v = e_j$ where $e_j$ is the $j$-th unit vector, we call $\pi'_{A,e_j,u}(\lambda)$ the \textit{j-th unit pseudo annihilating polynomial}, denoted as $\pi'_{A,j,u}(\lambda)$.

Unit pseudo annihilating polynomials can be calculated as follows. Let $u$ be a row vector of integers of dimension $n$, where $u = (u_1, \ldots, u_n)$ with $u_j \neq 0$ are randomly generated numbers for all $j$, and $w = (w_1, w_2, \ldots, w_j, \ldots, w_n) = up(A)$. Then, we have $u(p(A)e_j) = (up(A))e_j = we_j = w_j$, thus $w_j = 0$ if $p(A)e_j = 0$. Therefore, we see that $p(A)e_j \neq 0$ for every $j$ satisfying $w_j \neq 0$.

Let $u$ be defined as in the above, and let

$$w^{(0)}_p = (w^{(0)}_{p,1}, w^{(0)}_{p,2}, \ldots, w^{(0)}_{p,n}) = uG_p,$$

$$w^{(k)}_p = (w^{(k)}_{p,1}, w^{(k)}_{p,2}, \ldots, w^{(k)}_{p,n}) = uG_p F_p^k \text{ for } k > 0,$$  \hspace{1cm} (14)
where $G_p$ and $F_p$ are defined as in eq. (4). Furthermore, for $j = 1, \ldots, n$, define
\[
\rho_{p,j} = \begin{cases} 
0 & \text{if } w_{p,j}^{(0)} = 0, \\
k & \text{if } w_{p,j}^{(k-1)} \neq 0 \text{ and } w_{p,j}^{(k)} = 0.
\end{cases}
\] (15)

Then, we have the following lemma.

**Lemma 4.** For $\pi_{A,j}(\lambda)$ in eq. (3), $\rho_{p,j}$ in eq. (15), $1 \leq p \leq q$ and $j \in \{1, 2, \ldots, n\}$, we have $r_{j,p} \geq \rho_{p,j}$.

**Proof.** By the definition of $\pi_{A,j}(\lambda)$ and $g_p(\lambda)$, we have
\[
g_p(\lambda) = \left( \frac{1}{f_p(\lambda)^{r_{j,p}}} \right) \pi_{A,j}(\lambda) \bar{g}_{p,j}(\lambda),
\]
where
\[
\bar{g}_{p,j}(\lambda) = f_1(\lambda)^{m_1-r_{j,1}} \cdots f_{p-1}(\lambda)^{m_{p-1}-r_{j,p-1}} \times f_{p+1}(\lambda)^{m_{p+1}-r_{j,p+1}} \cdots f_q(\lambda)^{m_q-r_{j,q}}.
\]

Thus, for $u$ in eq. (14), we have
\[
w_{p,j}^{(r_{j,p})} = uG_pF_p^{r_{j,p}}e_j = u\bar{g}_{p,j}(A)\pi_{A,j}(A)e_j = u\bar{g}_{p,j}(A)0 = 0,
\]
which implies $r_{j,p} \geq \rho_{p,j}$. This completes the proof. $\square$

Lemma 4 tells us a way for calculating the $j$-th unit pseudo annihilating polynomial that is a factor of the $j$-th unit minimal annihilating polynomial. We summarize the steps for calculating the unit pseudo annihilating polynomials in Algorithm 4.

**Proposition 5.** For $A$ defined as in the above, Algorithm 4 outputs all unit pseudo annihilating polynomials of $A$ with
\[
O ( (q-1)n^3 + n^2 \deg(\pi_A(\lambda)))
\] (16)
arithmetic operations in $\mathbb{K}$. 

9
Algorithm 4 Calculating unit pseudo annihilating polynomials \( \pi'_{A,j,u}(\lambda) \)

**Input:** \( A \in K^{n \times n} \); \( \chi_A(\lambda) = f_1(\lambda)^{m_1} f_2(\lambda)^{m_2} \cdots f_q(\lambda)^{m_q} \in K[\lambda] \); \( \rho_{ij} \) is equal to exponent of factor \( f_i(\lambda) \) in the \( j \)-th unit pseudo annihilating polynomial \( \pi'_{A,j,u}(\lambda) \);

**Output:** \( P = (\rho_{ij}) \in \mathbb{R}^{q \times n} \), where \( \rho_{ij} \) is equal to exponent of factor \( f_i(\lambda) \) in the \( j \)-th unit pseudo annihilating polynomial \( \pi'_{A,j,u}(\lambda) \);

1: function \( \text{UNIT\_PSEUDO\_ANNIH}(A, \{ \{ f_1(\lambda), m_1 \}, \{ f_2(\lambda), m_2 \}, \ldots, \{ f_q(\lambda), m_q \} \}) \)
2: \( u \leftarrow (\text{a randomly generated row vector of dimension } n) \);
3: for \( i = 1, \ldots, q \) do
4: \( g_i(\lambda) \leftarrow \chi_A(\lambda)/(f_i(\lambda)^{m_i}) \);
5: \( \bar{b}_i = (\bar{b}_i^{(1)}, \ldots, \bar{b}_i^{(n)}) \leftarrow \text{VECTOR-MATRIX-HORNER}(g_i(\lambda), A, u) \); \( \bar{b}_i \leftarrow u g_i(A) \)
6: \( k \leftarrow 0 \);
7: \( \text{FL} = (\text{FL}_1, \text{FL}_2, \ldots, \text{FL}_n) = (0, \ldots, 0) \); \( \text{FL}_j = 1 \) if exponent \( \rho_{i,j} \) of \( f_i(\lambda) \) in the \( j \)-th unit pseudo annihilating polynomial \( \pi'_{A,j,u}(\lambda) \) has been discovered;
8: for \( l = 1, \ldots, m_i \) do
9: \( \text{if } \bar{b}_i = 0 \text{ and } \text{FL} = (1, \ldots, 1) \text{ then} \)
10: \( \text{break} \);
11: \( \text{end if} \)
12: for \( j = 1, \ldots, n \) do
13: \( \text{if } \bar{b}_i^{(j)} = 0 \text{ and } \text{FL}_j = 0 \text{ then} \) \( \rho_{i,j} \leftarrow k \); \( \text{FL}_j \leftarrow 1 \);
14: \( \text{end if} \)
15: \( \text{end for} \)
16: \( \bar{b}_i \leftarrow \text{VECTOR-MATRIX-HORNER}(f_i(\lambda), A, \bar{b}_i) \); \( k \leftarrow k + 1 \);
17: \( \text{end for} \)
Algorithm 4 Calculating unit pseudo annihilating polynomials $\pi'_{A,j,u}(\lambda)$ (Continued)

21: \hspace{1em} \textbf{for} $j = 1, \ldots, n$ \textbf{do}
22: \hspace{2em} \textbf{if} $\text{FL}_{ij} = 0$ \textbf{then} \hspace{1em} For $f_i(\lambda)$ of whose exponent in $\pi'_{A,j,u}(\lambda)$ has not found, it should be $m_i$;
23: \hspace{3em} $\rho_{i,j} \leftarrow m_i$;
24: \hspace{2em} \textbf{end if}
25: \hspace{1em} \textbf{end for}
26: \textbf{end for}
27: \textbf{return} $P = (\rho_{i,j})$; \hspace{1em} $\triangleright$ $\pi'_{A,j,u}(\lambda) = f_1(\lambda)^{\rho_{1,j}} f_2(\lambda)^{\rho_{2,j}} \cdots f_q(\lambda)^{\rho_{q,j}}$;
28: \textbf{end function}

Proof. By repeating the step in line 18 in Algorithm 4, we calculate $\rho_{i,j}$ in eq. (15). By Lemma 4, we see that $f_1(\lambda)^{\rho_{1,j}} f_2(\lambda)^{\rho_{2,j}} \cdots f_q(\lambda)^{\rho_{q,j}}$ with $\rho_{p,j}$ as in eq. (15) divides $\pi_{A,j}(\lambda)$ in eq. (3), thus we have $\pi'_{A,j,u}(\lambda) = f_1(\lambda)^{\rho_{1,j}} f_2(\lambda)^{\rho_{2,j}} \cdots f_q(\lambda)^{\rho_{q,j}}$ or the $j$-th pseudo unit minimal annihilating polynomial of $A$.

We estimate time complexity of the algorithm as follows. First, the amount of operations required for calculating $\bar{b}_i$ in lines 4 and 5 is estimated $O(n^2(n - d_i m_i))$, similarly as in eq. (10). Repeating this calculation for $i = 1, \ldots, q$ in the “for” loop from line 3 requires $O((q - 1)n^3)$ arithmetic operation by the same estimation as in eq. (11).

Next, in line 18 calculating $\bar{b}_i$ requires $O(n^2 d_i)$ operations. For each $i$ in the “for” loop in line 3 the “for” loop in line 8 repeats for $\max_{j \in \{1, \ldots, n\}} \rho_{i,j}$. Since $\rho_{i,j} \leq r_{j,i}$ and $\max_{j \in \{1, \ldots, n\}} r_{j,i} = l_i$, the number of operation is bounded to

$$O \left( n^2 \sum_{i=1}^{q} (l_i d_i) \right) = O(n^2 \deg(\pi_A(\lambda)))$$

for $i = 1, \ldots, q$ in the “for” loop in line 3. Sum of the amounts in eqs. (17) and (18) gives an estimate of the number of operations in the whole algorithm as in eq. (16), which proves the proposition. \qed
Remark 3. If we have the minimal polynomial along with its irreducible factorization as in eq. (5), time complexity of the algorithm becomes as
\[ O\left(q n^2 (\deg(\pi_A(\lambda)))\right), \]
(cf. eq. (10)) by \(n^3\) is replaced with \(n^2 (\deg(\pi_A(\lambda)))\) in eq. (17) for calculating \(b_i\) in lines 4 and 5 in Algorithm 4.

Remark 4. In Algorithm 4, each processes in the “for” loop in line 3 are independent each other thus we can execute them in parallel to make the calculation faster. For example, if we distribute each processes to \(M\) processors satisfying \(M \leq q\), the estimate of computing time in eq. (16) becomes
\[ O\left(\frac{q-1}{M} n^3 + \frac{n^2}{M} \deg(\pi_A(\lambda))\right). \]

With Algorithm 4, we define an algorithm for calculating the unit annihilating polynomials as in Algorithm 5. Then, we show the validity and the time complexity of the algorithm by the following propositions.

Proposition 6. For \(A\) defined as in the above, Algorithm 5 outputs the exponents of factors in the unit annihilating polynomials of \(A\).

Proof. For \(j = 1, \ldots, n\), let the \(j\)-th unit annihilating polynomial of \(A\) be
\[ \pi_{A,j}(\lambda) = f_1(\lambda)^{r_{1,j}} f_2(\lambda)^{r_{2,j}} \cdots f_q(\lambda)^{r_{q,j}}, \tag{19} \]
and the \(j\)-th pseudo unit annihilating polynomial of \(A\) with respect to \(u\) calculated by Algorithm 4 be
\[ \pi'_{A,j,u}(\lambda) = f_1(\lambda)^{\rho_{1,j}} f_2(\lambda)^{\rho_{2,j}} \cdots f_q(\lambda)^{\rho_{q,j}}. \tag{20} \]
We consider the following cases according to lines 6 and 7.

Case 1: \(v = 0\), which corresponds to \(\pi'_{A,j,u}(\lambda) = \pi_{A,j}(\lambda)\). In this case, the algorithm outputs \(\pi'_{A,j,u}(\lambda)\) as \(\pi_{A,j}(\lambda)\).

Case 2: \(v \neq 0\), which corresponds to \(\pi'_{A,j,u}(\lambda) \neq \pi_{A,j}(\lambda)\) and \(\deg(\pi'_{A,j,u}(\lambda)) < \deg(\pi_{A,j}(\lambda))\). For \(j = 1, \ldots, n\), \(r_{i,j}\) in eq. (19) and \(\rho_{i,j}\) in eq. (20), let
\[ q''_j \leq q \tag{21} \]
Algorithm 5 Calculating the unit minimal annihilating polynomials $\pi_{A,j}(\lambda)$

Input: $A \in K^{n \times n}$; \hspace{1cm} \triangleright \text{Input matrix;}

\[ \chi_A(\lambda) = f_1(\lambda)^{m_1} f_2(\lambda)^{m_2} \cdots f_q(\lambda)^{m_q} \in K[\lambda]; \] \hspace{1cm} \triangleright \text{Irreducible factorization of the characteristic polynomial of } A \text{ as in eq. (1);} 

Output: $R = (r_{i,j}) \in \mathbb{R}^{q \times n}$, where $r_{i,j}$ is equal to exponent of factor $f_i(\lambda)$ in the $j$-th unit minimal annihilating polynomial $\pi_{A,j}(\lambda)$;

1: function UNIT_MINIMAL_ANNIH($A, \{\{f_1(\lambda), m_1\}, \{f_2(\lambda), m_2\}, \ldots, \{f_q(\lambda), m_q\}\}$)

2: \hspace{1cm} $P \leftarrow$ UNIT_PSEUDO_ANNIH($A, \{\{f_1(\lambda), m_1\}, \{f_2(\lambda), m_2\}, \ldots, \{f_q(\lambda), m_q\}\}$);

\hspace{2cm} $\triangleright P = (\rho_{ij});$

3: \hspace{1cm} $R = (r_{i,j}) \leftarrow P;$

4: \hspace{1cm} for $j = 1, \ldots, n$ do

5: \hspace{2cm} $\pi'_{A,j,u}(\lambda) \leftarrow f_1(\lambda)^{\rho_{1j}} \cdots f_q(\lambda)^{\rho_{qj}};$ \hspace{1cm} $\triangleright \text{The } j\text{-th unit pseudo annihilating polynomial of } A;$

6: \hspace{2cm} $v \leftarrow \text{MATRIX-VECTOR-HORNER}(\pi'_{A,j,u}(\lambda), A, e_j);$ \hspace{1cm} $\triangleright$

\hspace{3cm} $v \leftarrow \pi'_{A,j,u}(A)e_j;$

7: \hspace{2cm} if $v = 0$ then

8: \hspace{3cm} continue;

9: \hspace{2cm} else

10: \hspace{3cm} $v_0 \leftarrow v;$

11: \hspace{3cm} for $i = 1, \ldots, q$ do

12: \hspace{4cm} for $l = 1, \ldots, m_i - \rho_{ij}$ do

13: \hspace{5cm} $v \leftarrow \text{MATRIX-VECTOR-HORNER}(f_i(\lambda), A, v);$ \hspace{1cm} $\triangleright$

\hspace{6cm} $v \leftarrow f_i(A)v$

14: \hspace{3cm} if $v = 0$ then

15: \hspace{4cm} $r_{ij} \leftarrow \rho_{ij} + l;$ \hspace{1cm} $\triangleright \text{Exponent of } f_i \text{ in the } j\text{-th unit minimal annihilating polynomial is equal to } \rho_{ij} + l;$

16: \hspace{4cm} $v_i \leftarrow 0;$

17: \hspace{4cm} for $k = i - 1, \ldots, 0$ do

18: \hspace{5cm} $v_k \leftarrow \text{MATRIX-VECTOR-HORNER}(f_i(\lambda)^k, A, v_k);$ \hspace{1cm} $\triangleright$

\hspace{6cm} $v_k \leftarrow f_i(A)^k v_k$

19: \hspace{2cm} end for

end for
Algorithm 5 Calculating the unit minimal annihilating polynomials $\pi_{A,j}(\lambda)$ (Continued)

20: break;
21: end if
22: end for
23: if $v = 0$ then
24: break;
25: else
26: $v_i \leftarrow v$;
27: end if
28: end for
29: for $k = i - 1, \ldots, 1$ do
30: $v \leftarrow v_{k-1}$;
31: for $l = 0, \ldots, m_k - \rho_{kj} - 1$ do
32: if $v = 0$ then
33: for $s = k - 2, \ldots, 0$ do
34: $v_s \leftarrow$ Matrix-vector-horner($f_k(\lambda)^l$, $A$, $v_s$); ▼
35: $v_s \leftarrow f_k(A)^l v_s$
36: break;
37: end if
38: $v \leftarrow$ Matrix-vector-horner($f_k(\lambda)$, $A$, $v$); ▼
39: $v \leftarrow f_k(A) v$
40: $r_{kj} \leftarrow r_{kj} + 1$;
41: end for
42: end if
43: end for
44: return $R = (r_{i,j})$; ▼ $\pi_{A,j}(\lambda) = f_1(\lambda)^{r_{1,j}} f_2(\lambda)^{r_{2,j}} \cdots f_q(\lambda)^{r_{q,j}}$;
45: end function
be the largest index of $i$ satisfying $\rho_{i,j} < r_{i,j}$ and let $\delta_{i,j} = r_{i,j} - \rho_{i,j}$.

For every $i$ in the “for” loop from line 11 and $l$ at the beginning of the “for” loop in line 12, we have

$$v = f_1(A)^{m_1} \cdots f_{i-1}(A)^{m_{i-1}} f_i(A)^{\rho_{i,j} - l - 1} f_{i+1}(A)^{m_{i+1,j}} \cdots f_q(A)^{\rho_q,j} e_j. $$

For making $v = 0$ in line 14 exponent of $f_i(A)$ must be greater than or equal to $r_{i,j}$ for $i = 1, \ldots, q''$. In fact, for the first time that line 14 is satisfied, we have $i = q''$ and $l = r_{q'',j} - \rho_{q'',j}$, and we have

$$v = 0 = f_1(A)^{m_1} \cdots f_{q''-1}(A)^{m_{q''-1}} f_{q''}(A)^{r_{q'',j} - 1} f_{q''+1}(A)^{\rho_{q'',+1,j}} \cdots f_q(A)^{\rho_q,j} e_j. $$

(22)

Then, by line 15 we have $r_{q'',j} \leftarrow \rho_{q'',j} + (r_{q'',j} - \rho_{q'',j}) = r_{q'',j}$.

At the end of “for” loop in line 28 for the $i$-th time, we have

$$v_s = f_1(A)^{m_1} \cdots f_s(A)^{m_s} f_{s+1}(A)^{\rho_{s+1,j}} \cdots f_q(A)^{\rho_q,j} e_j$$

for $s = 0, \ldots, i$ (note that we do not have factors $f_1(A)^{m_1} \cdots f_s(A)^{m_s}$ for $s = 0$).

Thus, when the line 14 is satisfied for $i = q''$, we have eq. (23) for $s = 0, \ldots, q'' - 1$. Then, by “for” loop between line 17 and 19 $v_s$ in eq. 23 gets updated as

$$v_s = f_1(A)^{m_1} \cdots f_s(A)^{m_s} f_{s+1}(A)^{\rho_{s+1,j}} \cdots f_{q''-1}(A)^{r_{q''-1,j}} \cdots f_{q''}(A)^{r_{q'',j}} f_{q''+1}(A)^{\rho_{q'',+1,j}} \cdots f_q(A)^{\rho_q,j} e_j,$$

for $s = 0, \ldots, q'' - 1$ (note that exponent of $f_{q''}(A)$ is equal to $r_{q'',j}$ which is equal to the one in the unit annihilating polynomial).

After exiting from “for” loop at line 28 we have $i = q''$, thus, at the first time for “for” loop in line 30 we have $k = q'' - 1$. For $k = q'' - 1, \ldots, 1$ in the “for” loop from line 30, we have

$$v = v_{k-1} = f_1(A)^{m_1} \cdots f_{k-1}(A)^{m_{k-1}} f_k(A)^{\rho_{k,j}} \cdots f_{k+1}(A)^{r_{k+1,j}} \cdots f_{q''}(A)^{r_{q'',j}} f_{q''+1}(A)^{\rho_{q'',+1,j}} \cdots f_q(A)^{\rho_q,j} e_j.$$
If \( \mathbf{v} \) satisfies line 32, then we have

\[
\mathbf{v} = 0 = f_1(A)^{m_1} \cdots f_{k-1}(A)^{m_{k-1}} f_k(A)^{r_{k,j}} \times f_{k+1}(A)^{r_{k+1,j}} \cdots f_{q''}(A)^{r_{q''-1,j}} f_{q''} (A)^{r_{q'',1,j}} \cdots f_q(A)^{r_{q,j}} e_j,
\]

with \( l = r_{k,j} - \rho_{k,j} \). Thus, by “for” loop between line 33 and 35, \( \mathbf{v} \) in eq. (23) gets updated as

\[
\mathbf{v}_s = f_1(A)^{m_1} \cdots f_s(A)^{m_s} f_{s+1}(A)^{\rho_{s+1,j}} \cdots f_{k-1}(A)^{\rho_{k-1,j}} \times f_k(A)^{r_{k,j}} \cdots f_{q''}(A)^{r_{q''-1,j}} f_{q''} (A)^{r_{q'',1,j}} \cdots f_q(A)^{r_{q,j}} e_j,
\]

for \( s = 0, \ldots, k - 1 \). Finally, for \( k = 1 \), we have eq. (23) as

\[
\mathbf{v}_0 = f_1(A)^{r_{1,j}} f_2(A)^{r_{2,j}} \cdots f_{q''}(A)^{r_{q''-1,j}} f_{q''} (A)^{r_{q'',1,j}} f_q(A)^{r_{q,j}} e_j,
\]

with \( l = r_{1,j} - \rho_{1,j} \). Thus, at exiting line 41 we have \( r_{i,j} \) satisfying \( \pi_{A,j}(\lambda) = f_1(\lambda)^{r_{1,j}} f_2(\lambda)^{r_{2,j}} \cdots f_q(\lambda)^{r_{q,j}} \) for each \( j \).

Remark 5. In Algorithm 5, in the case \( \mathbf{v} = \mathbf{0} \) in line 6, then the pseudo annihilating polynomial calculated in line 5 is true unit minimal annihilating polynomial, thus no more calculation is needed. On the other hand, in the case \( \mathbf{v} \neq \mathbf{0} \), the pseudo annihilating polynomial is a factor of true unit minimal annihilating polynomial and \( \mathbf{v} \) is a partial result, thus calculation of the unit minimal annihilating polynomial is accomplished by calculating the minimal annihilating polynomial of \( \mathbf{v} \) as in the rest of the algorithm. In this way, true unit annihilating polynomial is derived from the pseudo unit annihilating polynomial without restarting whole calculation, that makes proposed algorithm very efficient.

Proposition 7. For \( A \) defined as in the above, Algorithm 5 outputs the result with

\[
O \left( (q - 1)n^3 + n^2 \deg(\pi_A(\lambda)) + n^2 \sum_{j=1}^{n} \deg \pi'_{A,j}(\lambda) + n^2 \sum_{j=1}^{n} q''_j \sum_{k=1}^{n} d_k \{ (m_k - \rho_{kj}) + k\delta_{k,j} \} \right)
\]
arithmetic operations in \( \mathbb{K} \), where \( q''_j \leq q \) is the largest index of \( i \in \{1, \ldots, q\} \) satisfying \( \rho_{i,j} < r_{i,j} \) and \( \delta_{i,j} = r_{i,j} - \rho_{i,j} \) with \( j = 1, \ldots, n \), \( r_{i,j} \) as in eq. (19) and \( \rho_{i,j} \) as in eq. (20).

Proof. First, note that, line 2 can be executed with

\[
O \left( (q - 1)n^3 + n^2 \deg(\pi_A(\lambda)) \right)
\]

(26)
arithmetic operations as in eq. (16). Then, in the “for” loop in line 4 for \( j = 1, \ldots, n \), each loop has the following operations on vectors and matrices.

In line 6 calculating \( \nu \) costs \( O(n^2 \deg \pi'_{A,j}(\lambda)) \), which is bounded by \( O(n^2 \deg \pi_{A,j}(\lambda)) \). Thus, the number of arithmetic operation for this line is bounded by

\[
\sum_{j=1}^{n} O(n^2 \deg \pi_{A,j}(\lambda)),
\]

(27)
in the “for” loop in line 4.

In line 13 calculating \( \nu \) costs \( O(d_i n^2) \) for each \( i \) by the Horner’s rule, as shown in Proposition 1. Since this line is called for \( l = 1, \ldots, m_i \) (in line 12) with \( i = 1, \ldots, q''_j - 1 \) (in line 11) and \( l = 1, \ldots, \delta_{i,j} \) with \( i = q''_j \), the number of arithmetic operations for this line is bounded by

\[
\sum_{i=1}^{q''_j} O(d_i(m_i - \rho_{i,j})n^2).
\]

(28)

In line 18 calculating \( f_i(A)^l \nu_k \) costs \( O(d_{q''_j} \delta_{q''_j,j} n^2) \) by the Horner’s rule as in Proposition 1 since we have \( \deg((f_i)^l) = l \cdot \deg(f_i) = l \cdot d_i \) with \( i = q''_j \) and \( l = \delta_{q''_j,j} = r_{q''_j,j} - \rho_{q''_j,j} \). Since this line is called for \( k = i - 1, \ldots, 0 \) (in line 17) with \( i = q''_j \) (since this line is called when \( \nu \) satisfies line 14 that occurs only once for \( i = q''_j \); immediately after that there will be a break in line 24 of the “for” loop in \( i \) (in line 11)), the number of arithmetic operations for this line is bounded by

\[
O(q''_j d_{q''_j} \delta_{q''_j,j} n^2).
\]

(29)

In line 34 calculating \( f_k(A)^l \nu_s \) costs \( O(d_k \delta_{k,j} n^2) \) by the Horner’s rule as in Proposition 1 since we have \( \deg((f_k)^l) = l \cdot d_k \) with \( l = \delta_{k,j} \). Since this line
is called for \( k - 1 \) times (in the “for” loop in line 33) in each \( k \), the number of arithmetic operations for this line is bounded by \( O(kd_k \delta_{k,j} n^2) \) in that loop. Furthermore, this loop is called for \( k = i - 1, \ldots, 1 \) (in the “for” loop in line 29) with \( i = q''_j \), the number of arithmetic operations for this line is bounded by

\[
\sum_{k=1}^{q''_j - 1} O(kd_k \delta_{k,j} n^2).
\] (30)

In line 38 calculating \( f_k(A)v \) costs \( O(d_k n^2) \) by the Horner’s rule as in Proposition 1. Since this line is called for \( \delta_{k,j} \) times (for \( l = 0, \ldots, d_k - \rho_{k,j} - 1 \) in the “for” loop in line 31) in each \( k \) and for \( k = i - 1, \ldots, 1 \) (in the “for” loop in line 29) with \( i = q''_j \), the number of arithmetic operations for this line is bounded by

\[
\sum_{k=1}^{q''_j - 1} O(d_k \delta_{k,j} n^2).
\] (31)

In the above, we see that eqs. (29) and (30) are combined as

\[
\sum_{k=1}^{q''_j} O(kd_k \delta_{k,j} n^2),
\] (32)

which dominates eq. (31). By eqs. (28) and (32), we can estimate the number of arithmetic operations in the “for” loop in line 11 as

\[
\sum_{k=1}^{q''_j} O(d_k \{ (m_k - \rho_{kj}) + k \delta_{k,j} \} n^2),
\] (33)

which becomes as

\[
\sum_{j=1}^{n} \sum_{k=1}^{q''_j} O(d_k \{ (m_k - \rho_{kj}) + k \delta_{k,j} \} n^2),
\] (34)

for \( j = 1, \ldots, n \) (as in the “for” loop in line 11). Finally, by adding the result in eq. (34) together with the ones in eqs. (26) and (27), we have eq. (25), which proves the proposition.
Remark 6. If we have the minimal polynomial \( \pi_A(\lambda) \) along with its irreducible factorization as in eq. (5), time complexity of the algorithm becomes as

\[
O \left( qn^2 \left( \text{deg}(\pi_A(\lambda)) \right) + n^2 \sum_{j=1}^{n} \deg \pi'_{A,j}(\lambda) \right.
\]
\[
+ n^2 \sum_{j=1}^{n} \sum_{k=1}^{q'_{ij}} d_k \{ (l_k - \rho_{kj}) + k\delta_{k,j} \} \bigg),
\]

(cf. eq. (25)), by \((q - 1)n^3\) is replaced with \((q - 1)n^2(\text{deg}(\pi_A(\lambda)))\) and \(m_i\) is replaced with \(l_i\) in the proof as well as in Algorithm 5.

Remark 7. In Algorithm 5 each processes in the “for” loop in line 4 are independent each other thus we can execute them in parallel to make the calculation faster. For example, if we distribute each processes to \(M\) processors satisfying \(M \leq n\), the estimates of computing time in eq. (25) become as

\[
O \left( \frac{q - 1}{\min\{M, q\}} n^3 + \frac{n^2}{\min\{M, q\}} \text{deg}(\pi_A(\lambda)) + \frac{n^2}{M} \sum_{j=1}^{n} \deg \pi'_{A,j}(\lambda) \right.
\]
\[
+ \frac{n^2}{M} \sum_{j=1}^{n} \sum_{k=1}^{q'_{ij}} d_k \{ (m_k - \rho_{kj}) + k\delta_{k,j} \} \bigg),
\]

where first two terms are from Algorithm 4 and they have \(\min\{M, q\}\) in the denominators because the corresponding process can be distributed to at most \(q\) processors (see Remark 4).

Remark 8. With high possibility, unit pseudo annihilating polynomials calculated by Algorithm 4 are true unit minimal annihilating polynomials, calculated with time complexity dominated by \(O((q - 1)n^3)\) (see Proposition 5) which is more efficient than that of Algorithm 1 by a factor of \(O(n)\). Thus, in the most of cases, Algorithm 5 outputs all the unit annihilating polynomials of given matrix more efficiently than Algorithm 1.

Remark 9. With factorization of the characteristic polynomial and pseudo minimal polynomial are given, Algorithm 5 gives a practical method for calculating the minimal polynomial efficiently.
4. Efficient calculation of unit pseudo annihilating polynomials

Now, we propose an efficient method for calculating unit pseudo annihilating polynomials which is the main result of the present paper. For calculating \( \pi'_{A,j,u}(\lambda) \) for \( j = 1, 2, \ldots, n \) in Algorithm 3, \( q \) row vectors \( w_1^{(0)} = uG_1, w_2^{(0)} = uG_2, \ldots, w_q^{(0)} = uG_q \) of dimension \( n \) are needed, which are defined as

\[
\begin{align*}
    w_1^{(0)} &= uF_{m_2}^1 F_{m_3}^2 \cdots F_{m_q}^q, \\
    w_2^{(0)} &= uF_{m_2}^1 F_{m_3}^2 \cdots F_{m_q}^q, \\
    &\vdots \\
    w_q^{(0)} &= uF_{m_2}^1 F_{m_3}^2 \cdots F_{m_q}^q,
\end{align*}
\]

where \( F_i = f_i(A) \) for \( i = 1, \ldots, q \) and \( u \) is a random vector. Since \( G_1, \ldots, G_q \) consist of almost the same factors, \( w_1^{(0)}, \ldots, w_q^{(0)} \) are calculated efficiently with use of binary splitting technique, as follows.

**Example 1.** We show an example for \( q = 8 \). For simplicity, we denote a vector \( uF_{i_1}^{m_1} F_{i_2}^{m_2} \cdots F_{i_k}^{m_k} \) to \( u_{(i_1,i_2,\ldots,i_k)} \), for example, \( w_1^{(0)} = u_{(2,3,4,5,6,7,8)} \).

Since \( w_1^{(0)}, \ldots, w_4^{(0)} \) are calculated from \( G_1, \ldots, G_4 \), respectively, that have \( F_5^{m_5} F_6^{m_6} F_7^{m_7} F_8^{m_8} \) as a common factor, thus we first calculate \( u_{(5,6,7,8)} = uF_5^{m_5} F_6^{m_6} F_7^{m_7} F_8^{m_8} \).

Then, by multiplying \( F_3^{m_3} F_4^{m_4} \) and \( F_1^{m_1} F_2^{m_2} \), we have

\[
\begin{align*}
    u_{(3,4,5,6,7,8)} &= (uF_5^{m_5} F_6^{m_6} F_7^{m_7} F_8^{m_8})(F_3^{m_3} F_4^{m_4}), \\
    u_{(1,2,5,6,7,8)} &= (uF_5^{m_5} F_6^{m_6} F_7^{m_7} F_8^{m_8})(F_1^{m_1} F_2^{m_2}),
\end{align*}
\]

respectively. Furthermore, by multiplying \( u_{(3,4,5,6,7,8)} \) by \( F_2^{m_2} \) and \( F_1^{m_1} \) in eq. (36), we have \( u_{(2,3,4,5,6,7,8)} = w_1^{(0)} \) and \( u_{(1,3,4,5,6,7,8)} = w_2^{(0)} \), respectively, and by multiplying \( u_{(1,2,5,6,7,8)} \) by \( F_4^{m_4} \) and \( F_3^{m_3} \) in eq. (37), we obtain \( u_{(1,2,3,5,6,7,8)} = w_3^{(0)} \) and \( u_{(1,2,3,5,6,7,8)} = w_4^{(0)} \), respectively. With the same manner, \( u_{(1,2,3,4,6,7,8)} = w_5^{(0)} \), \( u_{(1,2,3,4,5,7,8)} = w_6^{(0)} \), \( u_{(1,2,3,4,5,6,7)} = w_7^{(0)} \), and \( u_{(1,2,3,4,5,6,7,8)} = w_8^{(0)} \) can be calculated as well, shown as a binary tree in Figure 4. In this example, \( \{w_1^{(0)}, w_2^{(0)}, w_3^{(0)}, w_4^{(0)}\} \) and \( \{w_5^{(0)}, w_6^{(0)}, w_7^{(0)}, w_8^{(0)}\} \) are calculated independently, thus these calculation can be parallelized.

We define the binary tree used in Example 4 as follows.
Figure 1: Calculating $w_j^{(0)}$s in eq. (35) with a binary tree. See Example 1 for details.

Figure 2: A binary tree corresponding to the one in Figure 1 calculated by Definition 3. See Example 2 for details.

**Definition 3** (A binary tree used to calculate $w_1^{(0)}, \ldots, w_q^{(0)}$ in eq. (35)). For $S = \{1, \ldots, q\}$, define a binary tree $(T_s, V_s)$, where $T_s$ is the set of the nodes and $V_s$ is the set of the vertices, satisfying the following conditions:

1. The root node is $S = \{1, \ldots, q\}$ and the child nodes are nonempty subset of $S$ of consecutive numbers;
2. The leaves consist of the singletons $\{1\}, \{2\}, \ldots, \{q\}$;
3. If $I = \{l, \ldots, m\}$ is a parent node and $J$ and $K$ are its left and right child nodes, respectively, then $J$ and $K$ satisfy

   $$J = \{l, \ldots, l + \lfloor (m - l)/2 \rfloor\}, \quad K = \{l + \lfloor (m - l)/2 \rfloor + 1, \ldots, m\}.$$ 

**Example 2.** A binary tree corresponding to the one in Example 1 (see Figure 1) is given according to Definition 3 as shown in Figure 2.
**Proposition 8.** Let $S = \{1, \ldots, q\}$ and $I$ be a node in the graph $(T_S, V_S)$ in Definition 3. For $I$, define a row vector $v_I$ as

$$v_I = u \prod_{j \in S \setminus I} F_{m_j}^j,$$

where $u$ and $F_j$ are the same as in eq. (35). Then, as in the leaves of the graph, we can calculate $w_1^{(0)}, \ldots, w_q^{(0)}$ in eq. (35) correctly. Furthermore, in calculation of $w_1^{(0)}, \ldots, w_q^{(0)}$, total number of multiplications of matrices $F_{m_p}^m$ to the right of a row vector $r$ is estimated as $O(q \log_2 q)$.

**Proof.** We show that tracing all the paths of $(T_S, V_S)$ from the root node to the leaves enables us to calculate $w_1^{(0)}, \ldots, w_q^{(0)}$. First, by eq. (38), we have $u = u \prod_{j \in S \setminus I} F_{m_j}^j = v_S$ and $w_p^{(0)} = u \prod_{j \in S \setminus \{p\}} F_{m_j}^j = v_{\{p\}}$ for $p = 1, \ldots, q$. Now, if $I$ is a parent node and $J$ is its child node, then we have a relationship between vectors $v_I$ and $v_J$ such that

$$v_J = v_I \prod_{j \in I \setminus J} F_{m_j}^j,$$

which implies that we can calculate a vector in the child node by using intermediate result in its parent node. For estimating total number of multiplications of matrices $F_{m_p}^m$, it requires $q$ multiplications of $F_{m_p}^m$ in total for calculating vectors in all the nodes of depth $k$, and the height of the tree is estimated as $O(\log_2 q)$, thus we have the claim. This completes the proof. \hfill \Box

**Corollary 9.** In Proposition 8, total number of arithmetic operations in $K$ for multiplications of matrices $F_p$ to the right of a row vector $r$ is estimated as $O(n^3 \log_2 q)$.

**Proof.** In Proposition 8 it seems that the number of arithmetic operations for calculating vectors in all the nodes of depth $k$ equals to $O(qM(n))$ for $q$ multiplications of $F_{m_p}^m$. However, it can be reduced to $O(n^3)$ with the Horner’s rule. Since the height of the tree is estimated as $O(\log_2 q)$, thus we have the claim. This completes the proof. \hfill \Box
With the binary splitting technique, algorithms for calculating unit pseudo annihilating polynomials as well as the unit minimal annihilating polynomials become more efficient.

**Proposition 10.** With the binary splitting as in Proposition 8, we can calculate unit pseudo annihilating polynomials by Algorithm 4 with

\[ O(n^3 \max\{1, \log_2 q\}) \]  

arithmetic operations in \( K \).

**Proof.** The number of arithmetic operations in Algorithm 4 is estimated as follows. First, in line 5 total number of operations for calculating \( \bar{b}_i \) for \( i = 1, \ldots, q \) is \( O(n^3 \log_2 q) \) as in Corollary 9. Next, in line 18 the number of operations for calculating \( \bar{b}_i \) is bounded above by \( O(n^2 d_i m_i) \) for each \( i \in \{1, \ldots, q\} \), thus total number of operations for \( i = 1, \ldots, q \) is bounded above by \( O(n^2 \sum_{i=1}^{q} (d_i m_i)) = O(n^3) \). As a consequence, total number of operations becomes as in eq. (39), which completes the proof.

**Theorem 11.** With the binary splitting as in Proposition 8 we can calculate the unit minimal annihilating polynomials by Algorithm 5 with

\[ O \left( n^3 \max\{1, \log_2 q\} + n^2 \deg(\pi_A(\lambda)) + n^2 \sum_{j=1}^{n} \deg \pi'_A,j(\lambda) \right. \] 

\[ + n^2 \sum_{j=1}^{n} \sum_{k=1}^{q'} d_k \left\{ (m_k - \rho_{kj}) + k\delta_{k,j} \right\} \]  

\[ n^2 \sum_{j=1}^{q''} \sum_{k=1}^{m_k - \rho_{kj}} \]  

arithmetic operations in \( K \).

**Proof.** In Proposition 7 now the first term in eq. (26) is replaced with eq. (39), thus eq. (25) is replaced with eq. (40), which proves the theorem.

**Remark 10.** If we have the irreducible factorization of the minimal polynomial
\( \pi_A(\lambda) \) as in eq. (5), eq. (40) becomes as

\[
O\left( n^2(\deg(\pi_A(\lambda))) \max\{1, \log_2 q\} + n^2 \deg(\pi_A(\lambda)) + n^2 \sum_{j=1}^{n} \deg\pi'_{A,j}(\lambda) \\
+ n^2 \sum_{j=1}^{n} \sum_{k=1}^{q''_j} d_k \{(l_k - \rho_{kj}) + k\delta_{kj}\} \right)
\]

by \( n^3 \) is replaced with \( n^2(\deg(\pi_A(\lambda))) \) and \( m_i \) is replaced with \( l_i \) as well as in Remark 11.

**Remark 11.** As well as in Remark 7 if we distribute each processes to \( M \) processors satisfying \( M \leq n \), the estimates of computing time in eq. (40) become as

\[
O\left( \frac{n^3}{\min\{M, q\}} \max\{1, \log_2 q\} + \frac{n^2}{\min\{M, q\}} \deg(\pi_A(\lambda)) + \frac{n^2}{M} \sum_{j=1}^{n} \deg\pi'_{A,j}(\lambda) \\
+ \frac{n^2}{M} \sum_{j=1}^{n} \sum_{k=1}^{q''_j} d_k \{(m_k - \rho_{kj}) + k\delta_{kj}\} \right).
\]

**Remark 12.** With factorization of the characteristic polynomial and pseudo minimal polynomial are given, the method in this section gives a efficient method for calculating the minimal polynomial (cf. Remark 9).

**5. Concluding remarks**

For given matrix over a field of characteristic zero, we have shown an efficient method for calculating the unit minimal annihilating polynomials via calculating pseudo annihilating polynomials. By time complexity analysis as described in our main theorem (Theorem 11), proposed algorithm outputs the unit minimal annihilating polynomials with arithmetic operations as in eq. (40). Furthermore, resulting algorithm has the following advantages.

1. While time complexity analysis of proposed algorithm is based on the number of arithmetic operations in \( \mathbb{K} \) in the present paper, time complexity of multi-precision integer arithmetic will become non-negligible in actual
calculation. Under this circumstances, using pseudo annihilating polynomials avoids a lot of unnecessary multi-precision integer calculations.

2. Pseudo annihilating polynomials are efficiently calculated by using random vectors (see Algorithm1). Furthermore, in the case that calculated pseudo annihilating polynomial is not true minimal annihilating polynomial, the true minimal annihilating polynomial is derived from already calculated results, which makes the proposed algorithm efficient.

3. Parallel processing can be applied to main blocks of algorithms (see Remarks 4, 7 and 11 for more efficient calculation.

In resulting algorithm, since pseudo annihilating polynomials are factors of the minimal annihilating polynomials, we take an approach first by calculating pseudo annihilating polynomials as candidates for the minimal annihilating polynomials by efficient method, then verifying its correctness. With high possibility, pseudo annihilating polynomials are true minimal annihilating polynomials, so we can calculate minimal annihilating polynomials in quite efficient way with this approach. This property is also used in other algorithms such as for calculating eigenvectors (19), (22), (23), (24), (25).

Since application of the unit minimal annihilating polynomials covers a variety of algorithms in (exact) numerical linear algebra, each application will be discussed in separate papers.

References

[1] J.-G. Dumas, T. Gautier, M. Giesbrecht, P. Giorgi, B. Hovinen, E. Kaltofen, B. D. Saunders, W. J. Turner, and G. Villard. LinBox: A generic library for exact linear algebra. In A. M. Cohen, X.-S. Gao, and N. Takayama, editors, ICMS 2002, Proceedings of the 2002 International Congress of Mathematical Software, Beijing, China, pages 40–50. World Scientific, 2002.

[2] Y. Iizuka and S. Tajima. An algorithm for spectral decomposition of matrices in the case that the minimal polynomial has more than one multiple
factors (in Japanese). In Computer Algebra — Design of Algorithms, Implementations and Applications, volume 1814 of RIMS Kōkyūroku, pages 9–16. Research Institute for Mathematical Sciences, Kyoto University, October 2012.

[3] K. Ohara and S. Tajima. Distributed computation of spectral decomposition and eigenvectors of matrices (in Japanese). In Computer algebra — Design of Algorithms, Implementations and Applications, volume 1666 of RIMS Kōkyūroku, pages 65–68. Research Institute for Mathematical Sciences, Kyoto University, October 2009.

[4] K. Ohara and S. Tajima. Spectral decomposition and eigenvectors of matrices by residue calculus. In The Joint Conference of ASCM 2009 and MACIS 2009: Asian Symposium of Mathematics; Mathematical Aspects of Computer and Information Sciences, volume 22 of COE Lecture Note, pages 137–140. Faculty of Mathematics, Kyushu University, 2009.

[5] K. Ohara and S. Tajima. Extended Horner’s rule and parallel algorithm for spectral decomposition of matrices and (in Japanese). In Computer Algebra: The Algorithms, Implementations and the Next Generation, volume 1785 of RIMS Kōkyūroku, pages 123–130. Research Institute for Mathematical Sciences, Kyoto University, March 2012.

[6] K. Ohara and S. Tajima. Parallelization of spectral decomposition of matrices using the minimal annihilating polynomials (in Japanese). In Computer Algebra — Design of Algorithms, Implementations and Applications, volume 1815 of RIMS Kōkyūroku, pages 21–28. Research Institute for Mathematical Sciences, Kyoto University, October 2012.

[7] K. Ohara and S. Tajima. Developing a software package for spectral decomposition of matrices on computer algebra system Risa/Asir (in Japanese). Sushiki-shori (Bulletin of Japan Society for Symbolic and Algebraic Computation), 19(2):59–67, 2013.
[8] K. Ohara and S. Tajima. On determining the structure of the invariant space of matrices via pseudo-annihilating polynomial (in Japanese). In *Developments in Computer Algebra Research and Collaboration with Industry*, volume 49 of *COE Lecture Note*, pages 113–118. Institute of Mathematics for Industry, Kyushu University, July 2013.

[9] K. Ohara and S. Tajima. Algorithms for calculating generalized eigenspaces using pseudo annihilating polynomials (in Japanese). In *Computer Algebra and Related Topics*, volume 1907 of *RIMS Kôkyûroku*, pages 62–70. Research Institute for Mathematical Sciences, Kyoto University, 2014.

[10] K. Ohara, S. Tajima, and A. Terui. Developing linear algebra packages on Risa/Asir for eigenproblems. In *Proceedings of the 4th International Congress on Mathematical Software (ICMS 2014)*, volume 8592 of *Lecture Notes in Computer Science*, pages 321–324. Springer, 2014.

[11] A. Storjohann. An $O(n^3)$ Algorithm for the Frobenius Normal Form. In *Proceedings of the 1998 International Symposium on Symbolic and Algebraic Computation*, ISSAC ’98, pages 101–105, 1998.

[12] S. Tajima. Residue analysis and spectral decomposition of matrices using differential operators (in Japanese). In *Computer Algebra — Design of Algorithms, Implementations and the Next Generation*, volume 1814 of *RIMS Kôkyûroku*, pages 17–28. Research Institute for Mathematical Sciences, Kyoto University, 2012.

[13] S. Tajima. Calculating generalized eigenspace of matrices (in Japanese). In *Computer Algebra: The Algorithms, Implementations and the Next Generation*, volume 1843 of *RIMS Kôkyûroku*, pages 146–154. Research Institute for Mathematical Sciences, Kyoto University, 2013.

[14] S. Tajima and M. Higuchi. Calculating eigenvectors of matrices using their resolvents (in Japanese). In *Computer algebra — Design of Algorithms, Implementations and Applications*, volume 1666 of *RIMS Kôkyûroku*, pages
57–64. Research Institute for Mathematical Sciences, Kyoto University, October 2009.

[15] S. Tajima and Y. Iizuka. On algorithms for spectral decomposition of matrices (in Japanese). In Computer algebra — Design of Algorithms, Implementations and Applications, volume 1666 of RIMS Kôkyûroku, pages 49–56. Research Institute for Mathematical Sciences, Kyoto University, October 2009.

[16] S. Tajima and K. Nara. Pseudo annihilating polynomials and their applications. In Computer Algebra — Design of Algorithms, Implementations and Applications, volume 1815 of RIMS Kôkyûroku, pages 1–12. Research Institute for Mathematical Sciences, Kyoto University, October 2012.

[17] S. Tajima, K. Nara, and K. Ohara. On calculating the minimal polynomial of matrices. In Computer Algebra — Design of Algorithms, Implementations and Applications, volume 1814 of RIMS Kôkyûroku, pages 1–8. Research Institute for Mathematical Sciences, Kyoto University, October 2012.

[18] S. Tajima, K. Ohara, and A. 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), volume 8592 of Lecture Notes in Computer Science, pages 346–351. Springer, 2014.

[19] S. Tajima, K. Ohara, and A. Terui. Improvement of efficiency of an algorithm for calculating eigenvectors of matrices with parallelized Horner’s rule for matrices (in Japanese). In Developments in Computer Algebra Research, volume 1976 of RIMS Kôkyûroku, pages 81–90. Research Institute for Mathematical Sciences, Kyoto University, 2015.

[20] S. Tajima, K. Ohara, and A. Terui. Calculating matrix inverse by the extended Horner’s rule with pseudo minimal polynomial II (in Japanese). In Developments in Computer Algebra — Recent Research and Re-Formation
of Basic Theory —, volume 2019 of RIMS Kôkyûroku, pages 28–38. Research Institute for Mathematical Sciences, Kyoto University, 2017.

[21] S. Tajima, K. Ohara, and A. Terui. Calculating matrix inverse by the extended Horner’s rule with pseudo minimal polynomial (in Japanese). Sushiki-shori (Bulletin of Japan Society for Symbolic and Algebraic Computation), in print.

[22] S. Tajima and A. Terui. Calculating eigenvectors of matrices using candidates for minimal annihilating polynomials II (in Japanese). In Developments in Computer Algebra Research and Collaboration with Industry, volume 49 of COE Lecture Note, pages 119–127. Institute of Mathematics for Industry, Kyushu University, July 2013.

[23] S. Tajima and A. Terui. Calculating eigenvectors of matrices using candidates for minimal annihilating polynomials III (in Japanese). In Computer Algebra and Related Topics, volume 1907 of RIMS Kôkyûroku, pages 50–61. Research Institute for Mathematical Sciences, Kyoto University, July 2014.

[24] S. Tajima and A. Terui. Calculating eigenvectors of matrices using pseudo minimal annihilating polynomials IV (in Japanese). In Computer Algebra and Related Topics, volume 1955 of RIMS Kôkyûroku, pages 188–197. Research Institute for Mathematical Sciences, Kyoto University, July 2015.

[25] A. Terui and S. Tajima. Calculating eigenvectors of matrices using candidates for minimal annihilating polynomials (in Japanese). In Computer Algebra — Design of Algorithms, Implementations and Applications, volume 1815 of RIMS Kôkyûroku, pages 13–20. Research Institute for Mathematical Sciences, Kyoto University, October 2012.