Electronic Research Archive

https://www.aimspress.com/journal/era

ERA, 33(9): 5252–5276.
DOI: 10.3934/era.2025235
Received: 13 July 2025
Revised: 25 August 2025
Accepted: 01 September 2025
Published: 05 September 2025

Research article

Numerical algorithm and complexity analysis for diagonalization of multivariate homogeneous polynomials

Lishan Fang, Hua-Lin Huang*and Yuechen Li

School of Mathematical Sciences, Huaqiao University, Quanzhou 362021, China

* **Correspondence:** Email: Hualin.Huang@hqu.edu.cn.

Abstract: We study the computational complexity of a diagonalization technique for multivariate homogeneous polynomials, expressing them as the sums of powers of independent linear forms. It is based on Harrison's center theory and consists of a criterion and a diagonalization algorithm. Detailed formulations and the computational complexity of each component of the technique are given. The complexity analysis focuses on the impacts of the number of variables and the degree of given polynomials. We show that this criterion runs in polynomial time, and the diagonalization process performs efficiently in numerical experiments. Other diagonalization techniques are reviewed and compared in terms of complexity.

Keywords: multivariate homogeneous polynomial; diagonalization; complexity analysis; center; symmetric tensor

1. Introduction

1.1. *Problem statement*

In this article, we are concerned about the diagonalization of any complex multivariate homogeneous polynomial $f(x_1, x_2, \dots, x_n) \in \mathbb{C}[x_1, x_2, \dots, x_n]$ of degree $d \geq 3$. A multivariate homogeneous polynomial f of degree d is called diagonalizable if, after a linear change of variables $x = Py$, it can be written as

$$f(x) = f(Py) = \lambda_1 y_1^d + \lambda_2 y_2^d + \cdots + \lambda_n y_n^d,$$

where $\lambda_i \in \mathbb{C}$. If this is possible, we determine the appropriate change of variables $x = Py$. This is a typical problem in classical invariant theory and has also attention from theoretical computer science and scientific computing.

1.2. Related work

The diagonalization of homogeneous polynomials has been studied by various researchers, covering both the criteria for diagonalizability and algorithms for diagonalization. Kayal [1] presented a criterion and algorithm based on polynomial factorization, which admits randomized polynomial time for real polynomials. Koiran [2] proposed a deterministic criterion for the diagonalizability of homogeneous polynomials with $d = 3$. Built on this criterion, a new randomized black-box criterion for f of $d > 3$ was given in [3]. In contrast, Robeva [4] addressed this problem from the perspective of symmetric tensors using the tensor power method. Some other studies are interested in the properties of diagonalization, such as orthogonality and unitarity [4–7].

Harrison [8] generalized Witt's algebraic theory of quadratic forms to higher-degree forms in the 1970s, which was later applied to decompose higher-degree forms [9, 10]. Huang et al. [9] presented a diagonalization technique based on Harrison's center theory and the properties of the sums of powers. It contains a criterion for diagonalizability and an iterative process to diagonalize f with an arbitrary n and d, which only requires us to solve systems of linear and quadratic equations. In addition, it determines whether f is orthogonally or unitarily diagonalized using Harrison's uniqueness of the decomposition.

1.3. Methods

The diagonalization techniques of homogeneous polynomials generally become computationally expensive as the dimensions or degrees of polynomials increase. Some [1, 4] rely on tasks such as polynomial factorization or the tensor power method, which become the computational bottleneck of their techniques when the given polynomials become complex [11]. Koiran and Skomra [2] provided a complexity analysis on a randomized and deterministic criterion, showing that their complexity is polynomially bounded. However, they may fail with a low probability, as shown in their error analysis.

In this article, we provide a complexity analysis of the center-based diagonalization algorithm in [9]. The criterion and diagonalization algorithm are divided into six steps, and their complexity is analyzed separately. Two examples are provided to illustrate the algorithm's performance. We also compare it with other techniques in terms of their computational complexity. Two numerical experiments are conducted to verify the theoretical analysis, and the results are provided in Appendices A and B.

1.4. Organization of the paper

The remainder of this paper is organized as follows. We show Harrison's center theory and a diagonalization algorithm in Section 2. The criterion of diagonalizability is described in Section 3. The diagonalization process and the determination of orthogonality and unitarity are discussed in Section 4. We compare its computational complexity with other diagonalization techniques in Section 5. Lastly, the major findings are summarized in Section 6.

2. Centers and diagonalization of homogeneous polynomials

2.1. Centers and the diagonalization criterion

We recall some necessary definitions of the center algebras of homogeneous polynomials below. More details are given in [9]. Let $f(x_1, x_2, \ldots, x_n) \in \mathbb{C}[x_1, x_2, \ldots, x_n]$ be a homogeneous polynomial of degree d. Its center $Z(f)$ is defined as

$$Z(f) := \left\{X \in \mathbb{C}^{n\times n} \mid (HX)^T = HX\right\}, \tag{2.1}$$

where $H = \left(\frac{\partial^2 f}{\partial x_i \partial x_j}\right)_{1\le i,j\le n}$ is the Hessian matrix of f.

Remark 2.1. *Consider the sum of the powers of variables $f(x_1, x_2, \ldots, x_n) = x_1^d + x_2^d + \cdots + x_n^d$. The Hessian matrix of f is diagonal and its ii-entry is $d(d-1)x_i^{d-2}$. Thus, the ij-entry of HX reads $d(d-1)x_i^{d-2}X_{ij}$ for $X = (X_{ij})_{1\le i,j\le n}$. Since HX is symmetric, we can see that $Z(f)$ consists of all diagonal matrices. Thus, $Z(f)$ is a commutative associative subalgebra of the full matrix algebra and is isomorphic to $\mathbb{C}^n$ as algebras.*

A homogeneous polynomial f is said to be nondegenerate if none of its variables can be removed by an invertible linear change of variables. In other words, f is degenerate if and only if a change of variables $x = Py$ exists such that

$$g(y_1, y_2, \ldots, y_n) := f(Py) = g(y_1, y_2, \ldots, y_{n-1}, 0),$$

which implies that $\frac{\partial g}{\partial y_n} = 0$. We also have $\frac{\partial g}{\partial y_n} = \sum_{i=1}^n \frac{\partial f}{\partial x_i}\frac{\partial x_i}{\partial y_n}$ by the chain rule of derivatives. Therefore, f is degenerate if and only if its first-order differentials $\frac{\partial f}{\partial x_i}$ are linearly dependent. Additionally, any f can be reduced to a nondegenerate one with a suitable number of variables.

Proposition 2.2. *Let $f(x_1, x_2, \ldots, x_n)$ be as above and assume* $\operatorname{Rank}\left\{\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}\right\} = r$. *Then a change of variables $x = Py$ exists such that*

$$g(y_1, y_2, \ldots, y_n) := f(Py) = g(y_1, y_2, \ldots, y_r, 0, \ldots, 0).$$

Moreover, if we remove the redundant variables from $g(y)$ and view it as an r-variate polynomial, then it becomes nondegenerate.

Proof. Without loss of generality, we may assume that $\left\{\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_r}\right\}$ are linearly independent and

$$\frac{\partial f}{\partial x_j} = \sum_{i=1}^{r} \lambda_{j,i}\frac{\partial f}{\partial x_i} \qquad \text{for } j = r+1, \ldots, n,$$

where $\lambda_{j,i} \in \mathbb{C}$. Let P take the form of

$$P = \begin{pmatrix} I_r & \Lambda \\ \mathbf{0} & -I_{n-r} \end{pmatrix},$$

where I_r is a $r \times r$ identity matrix and

$$\Lambda = \begin{pmatrix} \lambda_{r+1,1} & \lambda_{r+1,2} & \cdots & \lambda_{r+1,r} \\ \lambda_{r+2,1} & \lambda_{r+2,2} & \cdots & \lambda_{r+2,r} \\ \vdots & \vdots & & \vdots \\ \lambda_{n,1} & \lambda_{n,2} & \cdots & \lambda_{n,r} \end{pmatrix}.$$

Take the change of variables $x = Py$ such that $g(y) = f(Py)$. Then we have

$$\frac{\partial g}{\partial y_j} = \frac{\partial f}{\partial x_j} \qquad \text{for } 1 \le j \le r,$$

$$\frac{\partial g}{\partial y_j} = \sum_{i=1}^{r} \frac{\partial f}{\partial x_i}\frac{\partial x_i}{\partial y_j} + \sum_{i=r+1}^{n} \frac{\partial f}{\partial x_i}\frac{\partial x_i}{\partial y_j} = \sum_{i=1}^{r} \lambda_{j,i}\frac{\partial f}{\partial x_i} - \frac{\partial f}{\partial x_j} = 0 \qquad \text{for } j > r.$$

That is to say, the variables $y_{r+1}, \dots, y_n$ do not occur in $g(y)$, and we may view g as an r-variate polynomial. In addition, g is nondegenerate in $\mathbb{C}[y_1, y_2, \dots, y_r]$ as $\left\{\frac{\partial g}{\partial y_1}, \frac{\partial g}{\partial y_2}, \dots, \frac{\partial g}{\partial y_r}\right\}$ are linearly independent. □

With the proposition above, in the following, there is no harm in considering only nondegenerate polynomials for our purpose. We can use the center in the criterion to determine whether a homogeneous polynomial f is diagonalizable. The following proposition was essentially obtained in [8]; see also [9, 10, 12].

Proposition 2.3. *Suppose that $f(x_1, x_2, \dots, x_n) \in \mathbb{C}[x_1, x_2, \dots, x_n]$ is a nondegenerate homogeneous polynomial of degree $d \ge 3$. Then f is diagonalizable if and only if $Z(f) \cong \mathbb{C}^n$ as algebras.*

Proof. If f is nondegenerate and diagonalizable, then there is a change of variables $x = Py$ such that

$$g(y) := f(Py) = \lambda_1 y_1^d + \lambda_2 y_2^d + \cdots + \lambda_n y_n^d,$$

where $\prod_{i=1}^n \lambda_i \ne 0$. Let $G = \left(\frac{\partial^2 g}{\partial y_i \partial y_j}\right)_{1\le i,j\le n}$ denote the Hessian matrix of g with respect to the variables y_i. Then by the chain rule of derivatives, it is easy to see that $G = P^T H P$. Thus, we have

$$(GY)^T = GY \Leftrightarrow (HPYP^{-1})^T = HPYP^{-1}.$$

It follows that $Z(f) = PZ(g)P^{-1}$. According to Remark 2.1, it is ready to see that $Z(g)$ consists of all diagonal matrices. Hence $Z(f)$ is isomorphic to $\mathbb{C}^n$ as algebras.

Conversely, if $Z(f) \cong \mathbb{C}^n$, then $Z(f)$ has a basis consisting of orthogonal primitive idempotents, say $\epsilon_1, \epsilon_2, \dots, \epsilon_n$. Since $\epsilon_1, \epsilon_2, \dots, \epsilon_n$ mutually commute and are diagonalizable, there is an invertible matrix P such that $P^{-1}\epsilon_i P = E_i$ for all i, where E_i's ii-entry is 1 and 0 elsewhere. Take the change of variables $x = Py$, in this case, it is not hard to verify that $g(y)$ is diagonal. Indeed, by the argument above, we have $Z(g) = P^{-1}Z(f)P$, and $Z(g)$ consists of all diagonal matrices. Therefore, G must be diagonal due to the condition of center algebras. This happens only if g is diagonal and hence f is diagonalizable. This proves the statement. □

It is also of interest whether a polynomial f is orthogonally or unitarily diagonalizable, namely, whether a change of variables $x = Py$ exists, in which P is orthogonal or unitary so that $f(Py)$ is diagonal [4–7]. The previous proposition can be applied to this problem effectively, as Harrison states that the diagonalization of a diagonalizable polynomial is essentially unique [8]. This is also obvious from the fact that a finite-dimensional commutative algebra has a unique set of orthogonal primitive idempotents disregarding the order; see [13]. Thus, we can determine whether f is orthogonally or unitarily diagonalizable by directly checking P.

2.2. *Diagonalization algorithm*

We show an algorithm to diagonalize the given homogeneous polynomials in Algorithm 1. It consists of two phases: Determining whether a polynomial f is diagonalizable in Steps 1 to 3. And an iterative process for diagonalizing f in Steps 4 to 9. We divide this algorithm into six compute the idempotents, diagonalize the form, and detect orthogonality, corresponding to Steps 1, 2, 3, 6, 7, and 9, respectively. Note that we also need to compute centers in Step 5 for the decomposed form of f. Detailed descriptions, a complexity analysis, and examples are given in Sections 3 and 4.

Algorithm 1 Diagonalization of homogeneous multivariate polynomials

```
    Input: polynomial $f$
    Output: diagonal form of $f$
1:  Determine linear independence of $\left\{\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n}\right\}$. If they are linearly independent, $f$ is
    nondegenerate; otherwise, transform $f$ to a nondegenerate form with fewer variables according
    to Proposition 2.2.
2:  Solve the system $(HX)^T = HX$ to compute the center $Z(f)$. If $\dim Z(f) \neq n$, reject it as $f$ is not
    diagonalizable.
3:  Use the Euclid algorithm to determine whether the minimal polynomials of the basis matrices
    $X_1, X_2, \dots, X_n$ of $Z(f)$ have multiple roots. If any multiple roots are found, reject the result, as $f$
    is not diagonalizable.
4:  while $f$ is not yet in a diagonal form do
5:      Solve the system $(HX)^T = HX$ to compute the center $Z(f)$.
6:      Compute orthogonal idempotents $(\epsilon, I - \epsilon)$ using $Z(f)$ with $\operatorname{Rank} \epsilon = 1$ and $\operatorname{trace}(\epsilon) = 1$
        conditions.
7:      Decompose the form of $f$ using the change of variables $x = Py$ and separate disjoint
        variables.
8:  end while
9:  Determine the orthogonality and unitarity of diagonalization on the basis of $P$.
```

The complexity analysis in this article focuses on the theoretical upper bounds of the computational costs in terms of three factors: Dimension n, degree d, and the number of monomial terms t of f. As n, d, and t increase, the costs of the diagonalization algorithm rise. The values of n and d are independent in f, and t increases if either n or d increases with an upper bound $\binom{n+d-1}{d}$. While the values of t increase exponentially for a large n and d, the growth is only polynomial if either n or d is fixed. In addition, t does not affect the dimensions of the two systems of equations in Algorithm 1. Discussions of the growth of t are provided in Appendix B.

3. Criterion of diagonalizability

The criterion for determining the diagonalizability of a given f is described in three steps: Detect nondegeneracy, compute the center, and detect diagonalizability. We provide their formulations and a complexity analysis in Subsections 3.1–3.3. It is shown that this criterion has polynomial runtime. In addition, two examples are given to demonstrate the procedure in Subsection 3.4.

3.1. Detect nondegeneracy

The process begins by detecting and reducing the nondegeneracy of f before the diagonalization, as shown in Algorithm 2. Recall that f is nondegenerate if and only if its first-order differentials $\frac{\partial f}{\partial x_i}$ are linearly independent, as discussed in Proposition 2.2. The linear dependence of $\frac{\partial f}{\partial x_i}$ for $i = 1, \ldots, n$ is determined by transforming them to an $n \times t_{max}$ matrix A, where its ij-th entry A_{ij} equals the coefficient of $\frac{\partial f_{(j)}}{\partial x_i}$, $f_{(j)}$ is the j-th term of f and $t_{max} = \binom{n+d-1}{d}$. Here and below, the terms of the monomials of f are lexicographically ordered. We then have

$$\operatorname{Rank} A = \operatorname{Rank}\left\{\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots, \frac{\partial f}{\partial x_n}\right\}.$$

If $\operatorname{Rank} A = n$, then $\frac{\partial f}{\partial x_i}$ are linearly independent and f is nondegenerate. Otherwise, f is degenerate, and we reduce f into a nondegenerate form with fewer variables.

Algorithm 2 Detect and reduce the nondegeneracy of polynomials

```
   Input: polynomial $f$
   Output: nondegenerate form of $f$
1: Calculate $\frac{\partial f}{\partial x_i}$ for $i = 1, \ldots, n$ and build matrix $A$.
2: if Rank $A < n$ then
3:     Compute $r$ maximum linearly independent subset $\left\{\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_r}\right\}$ on the basis of $A$.
4:     Reduce $f$ into a nondegenerate form with $r$ variables.
5: end if
```

We compute the reduced row echelon form (RREF) of A using Gaussian elimination to identify the maximum linearly independent subset $\left\{\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_r}\right\}$ with the rank $r < n$. The other dependent differentials can be expressed as

$$\frac{\partial f}{\partial x_j} = \sum_{i=1}^{r} \lambda_{j,i} \frac{\partial f}{\partial x_i} \qquad \text{for } j = r+1, \ldots, n.$$

We obtain an invertible matrix P as in Proposition 2.2 and a nondegenerate polynomial $g = f(Py)$ in r variables. Note that we replace g and r with f and n, respectively, in the rest of the article for the nondegenerate form, and the original n is represented as $\bar{n}$.

The construction of A involves the calculation of derivatives $\frac{\partial f}{\partial x_i}$, which takes $O(n^2 t)$ arithmetic operations. Rank A is calculated by computing the RREF of A using Gaussian elimination with $O(n^3)$ arithmetic operations [14]. While faster algorithms for rank calculation like those in [15] achieve $O(nt)$ complexity for A with $n < t$, the RREF of A is also required for reducing the degeneracy and prefered. Therefore, the complexity of detecting the nondegeneracy of f is $O(n^2 t)$.

Reducing the degeneracy of f takes $O(nd)$ operations to reduce one variable by a change of variables. It also needs to expand at most d terms of f in this process, which involves numerous multivariate polynomial multiplications. Since multiplying two polynomials takes $O(n \log n \log\log n)$ operations for polynomials with $d < n$, this process takes at most $O(n^2 d \log n \log\log n)$ operations [16]. Thus, we deduce that the overall computational complexity of the step of detecting nondegeneracy is $O(n^2 t)$.

Proposition 3.1. *Detecting and reducing the nondegeneracy of f requires $O(n^2 t)$ arithmetic operations.*

3.2. Compute the center

We continue to compute the center $Z(f)$ of a nondegenerate form f, which is the core of this diagonalization algorithm. This step, as described in Algorithm 3, is called at most $n-1$ times during the iterative process.

Algorithm 3 Compute the center of polynomials

```
    Input: polynomial $f$
    Output: Center $Z(f)$ and basis matrices $X_1, X_2, \dots, X_n$
1:  Compute the Hessian matrix $H$ of $f$ and build a system of equations $CX_v = 0$ equivalent to
    $(HX)^T = HX$.
2:  Compute the RREF of $C$ and obtain $Z(f)$. If $\dim Z(f) \neq n$, reject the result as $f$ is not
    diagonalizable.
3:  Obtain the basis matrices $X_1, X_2, \dots, X_n$ from $Z(f)$.
```

Recall that center $Z(f)$ is computed by solving the matrix equation $(HX)^T = HX$ in Eq (2.1), which can be transformed into a system of linear equations. Given an $n \times n$ matrix $X = \left(x_{ij}\right)_{1 \le i,j \le n}$, let X_v be the n^2-dimensional column vector

$$\begin{pmatrix} X_{(1)} \\ \vdots \\ X_{(n)} \end{pmatrix},$$

where $X_{(i)}$ is the i-th column of X. Let X_v^T denote the corresponding vector of X^T and let P_C be the permutation matrix such that $X_v^T = P_C X_v$. We denote M_{d-2} as the set of monomials of degree $d-2$ and express the Hessian matrix H as $H = \sum_{m \in M_{d-2}} mH_m$. Then $(HX)^T = HX$ can be rewritten as a system of linear equations

$$CX_v = (I_n \otimes H_m - (H_m \otimes I_n)\, P_C)\, X_v = 0, \quad m \in M_{d-2},$$

where $\otimes$ is the tensor product for multilinear algebra; see [10] for more details. We obtain $Z(f)$ by solving $CX_v = 0$ using Gaussian elimination and computing the RREF C_R. The resulting column vector X_v is transformed back to an $n \times n$ matrix X. If $\dim Z(f) = n$, the basis matrices $X_1, X_2, \dots, X_n$ of $Z(f)$ are chosen for each free variable in C_R by identifying the linear dependence between the variables. If $\dim Z(f) \neq n$, we deduce that f is not diagonalizable.

A significant portion of the computational cost in this step involves constructing the matrix C and calculating C_R. We build C by iterating through $(I_n \otimes H_m - (H_m \otimes I_n)\, P_C)_{m \in M_{d-2}}$ and the process takes at most $O(n^3 d^3 t)$ arithmetic operations. The dimension of the resulting matrix C is at most $n^d \times n^2$ with $\operatorname{Rank} C = n^2 - n$ [9, 10]. Consequently, it contains many redundant equations and zero entries. For instance, the dimension of C in Example 3.4 becomes 261×16 after removing redundant rows, which is markedly smaller than the theoretical upper bound of 1024×16. Thus, C can be preprocessed and then reduced to C_R using Gaussian elimination, which requires $O(n^6)$ operations [17]. More statistics are given in Table A2, which shows that the dimension of C increases with the order $O(n^{4.50})$. Appendix A also demonstrates that the time for solving $CX_v = 0$ increases with the order $O(n^{6.66})$ for $n \le 7$. Lastly, the costs of building H and choosing $X_1, X_2, \dots, X_n$ are trivial, and the overall complexity of computing center $Z(f)$ is $O(n^3 d^3 t + n^6)$.

Proposition 3.2. *Computing the center $Z(f)$ takes at most $O(n^3d^3t + n^6)$ arithmetic operations.*

3.3. Detect diagonalizability

The diagonalizability of f is determined using the semisimplicity of $Z(f)$'s basis matrices $X_1, X_2, \ldots, X_n$ as shown in Algorithm 4. Minimal polynomial $p_i(\lambda)$ for each X_i is built by iteratively testing the linear dependence of its powers $X_i^1, X_i^2, \ldots, X_i^j$ for $j \le n$. Let the matrix D of size $n^2 \times (j + 1)$ take the form of

$$D = \begin{pmatrix} I_{n,v} & D_{1,v} & \cdots & D_{j,v} \end{pmatrix},$$

where $I_{n,v}$ and $D_{j,v}$ are n^2-dimensional column vectors of the matrices I_n and X_i^j, respectively. They are defined in a similar way to X_v of X in Section 3.2. This process takes at most $n - 1$ rounds of the matrix multiplications and at most $O(n^4)$ operations in total. If $\operatorname{Rank} D < j$, $X_i^1, X_i^2, \ldots, X_i^j$ are linearly dependent. The corresponding minimal polynomial p_i is represented as

$$p_i(\lambda) = \sum_{j=0}^{m_i} t_j \lambda^j,$$

where t_j is the leading coefficient of the j-th row of the RREF of D. Compute the greatest common divisor (GCD) of $p_i(\lambda)$ and $p_i'(\lambda)$ using the Euclid algorithm, which takes $O(n^2)$ operations [18]. If the $\mathrm{GCD}\left(p_i(\lambda), p_i'(\lambda)\right) = 1$, $p_i(\lambda)$ does not have multiple roots. If the GCD of all basis matrices X_i is 1, we deduce that f is diagonalizable.

Algorithm 4 Detect the diagonalizability of polynomials

```
   Input: basis matrices $X_1, X_2, \ldots, X_n$
   Output: diagonalizability of $f$
1: for each basis matrix $X_i$ of $Z(f)$ do
2:     for $j = 1, \ldots, n$ do
3:         Build the matrix $D$ containing coefficients of $X_i^1, X_i^2, \ldots, X_i^j$.
4:         if $X_i^1, X_i^2, \ldots, X_i^j$ are linearly dependent then
5:             Build the minimal polynomial $p_i(\lambda)$. If $GCD\left(p_i(\lambda), p_i'(\lambda)\right) \neq 1$, $p_i(\lambda)$, $f$ is not diagonalizable.
6:         end if
7:     end for
8: end for
```

Algorithm 4 contains a nested loop for each basis matrix X_i, which computes the linear dependence for at most the n-th power of X_i. Each inner loop consists of a matrix multiplication, a rank calculation, and the Euclid algorithm. Therefore, the overall computational complexity for determining diagonalizability is $O(n^5)$.

Proposition 3.3. *Detecting the diagonalizability of f has a computational complexity of $O(n^5)$.*

In summary, the criterion for diagonalizability involves detecting the nondegeneracy, compute the center, and detecting the diagonalizability steps with a complexity of $O(n^2t)$, $O(n^3d^3t + n^6)$, and $O(n^5)$, respectively. The computation of the center $Z(f)$ is the most computationally expensive part of the criterion, and the overall complexity of the criterion is $O(n^3d^3t + n^6)$.

3.4. Some examples

Example 3.4. *Consider the following quadruple quintic form:*

$$\begin{aligned} f = {} & -1023x_1^5 + 2720x_1^4x_2 - 5200x_1^4x_3 - 2725x_1^4x_4 - 2240x_1^3x_2^2 + 9600x_1^3x_2x_3 + 3840x_1^3x_2x_4 - 10000x_1^3x_3^2 \\ & - 9280x_1^3x_3x_4 - 1590x_1^3x_4^2 + 1600x_1^2x_2^3 - 8640x_1^2x_2^2x_3 - 5760x_1^2x_2^2x_4 + 16320x_1^2x_2x_3^2 + 19200x_1^2x_2x_3x_4 \\ & + 7680x_1^2x_2x_4^2 - 10520x_1^2x_3^3 - 17040x_1^2x_3^2x_4 - 11040x_1^2x_3x_4^2 - 3530x_1^2x_4^3 - 160x_1x_2^4 + 1920x_1x_2^3x_3 \\ & - 6720x_1x_2^2x_3^2 - 3840x_1x_2^2x_3x_4 + 1920x_1x_2^2x_4^2 + 9600x_1x_2x_3^3 + 11520x_1x_2x_3^2x_4 - 3840x_1x_2x_4^3 \\ & - 4970x_1x_3^4 - 9040x_1x_3^3x_4 - 4080x_1x_3^2x_4^2 + 2240x_1x_3x_4^3 + 2085x_1x_4^4 + 64x_2^5 - 480x_2^4x_3 - 480x_2^4x_4 \\ & + 1600x_2^3x_3^2 + 2560x_2^3x_3x_4 + 1600x_2^3x_4^2 - 2880x_2^2x_3^3 - 5760x_2^2x_3^2x_4 - 5760x_2^2x_3x_4^2 - 2880x_2^2x_4^3 + 2720x_2x_3^4 \\ & + 6400x_2x_3^3x_4 + 7680x_2x_3^2x_4^2 + 6400x_2x_3x_4^3 + 2720x_2x_4^4 - 1055x_3^5 - 2870x_3^4x_4 - 3800x_3^3x_4^2 - 3760x_3^2x_4^3 \\ & - 2800x_3x_4^4 - 1025x_4^5 \end{aligned}$$

in $\mathbb{C}[x_1, x_2, x_3, x_4]$. *Calculate its partial derivatives* $\frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_4}$ *and build the matrix A. The calculation gives* $\operatorname{Rank} A = 4$, *which shows that* $\frac{\partial f}{\partial x_i}$ *are linearly independent and* f *is nondegenerate. Compute* $Z(f)$ *according to Eq* (2.1)*, and a general solution reads* $X = \lambda_1 X_1 + \lambda_2 X_2 + \lambda_3 X_3 + \lambda_4 X_4$*, where*

$$X_1 = \begin{pmatrix} 8 & 0 & 0 & -3 \\ 3 & 5 & 0 & -3 \\ -4 & 0 & 5 & 4 \\ 5 & 0 & 0 & 0 \end{pmatrix}, \quad X_2 = \begin{pmatrix} 3 & 5 & 0 & -3 \\ 28 & 20 & -20 & -43 \\ 16 & 0 & -5 & -16 \\ 0 & 5 & 0 & 0 \end{pmatrix},$$

$$X_3 = \begin{pmatrix} -4 & 0 & 0 & 4 \\ 16 & 0 & -5 & -16 \\ 12 & 0 & -10 & -12 \\ 5 & 0 & 0 & 0 \end{pmatrix}, \quad X_4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

The minimal polynomials $p_i(\lambda)$ *of* $X_1, \dots, X_4$ *are* $\lambda^2 - 1.6\lambda + 0.6$, $\lambda^4 - 3.6\lambda^3 + 0.8\lambda^2 + 3.6\lambda - 1.8$, $\lambda^3 + 2.8\lambda^2 + 1.6\lambda$, *and* $\lambda - 1$, *respectively. Since* $GCD\left(p_i(\lambda), p_i'(\lambda)\right) = 1$ *for* $i = 1, \dots, 4$, *we deduce that* f *is diagonalizable.*

Example 3.5. *Consider the following sextuple septic form:*

$$\begin{aligned} f = {} & 128x_1^7 + 448x_1^6x_2 - 7x_1^6x_3 + 14x_1^6x_5 + 672x_1^5x_2^2 - 21x_1^5x_3^2 + 84x_1^5x_5^2 + 560x_1^4x_2^3 - 35x_1^4x_3^3 + 280x_1^4x_5^3 \\ & + 280x_1^3x_2^4 - 35x_1^3x_3^4 + 560x_1^3x_5^4 + 84x_1^2x_2^5 - 21x_1^2x_3^5 + 672x_1^2x_5^5 + 14x_1x_2^6 - 7x_1x_3^6 + 448x_1x_5^6 + x_2^7 \\ & - 7x_3^6x_4 - 21x_3^6x_6 + 21x_3^5x_4^2 + 126x_3^5x_4x_6 + 189x_3^5x_6^2 - 35x_3^4x_4^3 - 315x_3^4x_4^2x_6 - 945x_3^4x_4x_6^2 - 945x_3^4x_6^3 \\ & + 35x_3^3x_4^4 + 420x_3^3x_4^3x_6 + 1890x_3^3x_4^2x_6^2 + 3780x_3^3x_4x_6^3 + 2835x_3^3x_6^4 - 21x_3^2x_4^5 - 315x_3^2x_4^4x_6 - 1890x_3^2x_4^3x_6^2 \\ & - 5670x_3^2x_4^2x_6^3 - 8505x_3^2x_4x_6^4 - 5103x_3^2x_6^5 + 7x_3x_4^6 + 126x_3x_4^5x_6 + 945x_3x_4^4x_6^2 + 3780x_3x_4^3x_6^3 \\ & + 8505x_3x_4^2x_6^4 + 10206x_3x_4x_6^5 + 5103x_3x_6^6 - 2x_4^7 + 14x_4^6x_5 - 21x_4^6x_6 - 84x_4^5x_5^2 - 189x_4^5x_6^2 + 280x_4^4x_5^3 \\ & - 945x_4^4x_6^3 - 560x_4^3x_5^4 - 2835x_4^3x_6^4 + 672x_4^2x_5^5 - 5103x_4^2x_6^5 - 448x_4x_5^6 - 5103x_4x_6^6 + 2443x_5^7 \\ & + 10206x_5^6x_6 + 20412x_5^5x_6^2 + 22680x_5^4x_6^3 + 15120x_5^3x_6^4 + 6048x_5^2x_6^5 + 1344x_5x_6^6 - 2059x_6^7 \end{aligned}$$

in $\mathbb{C}[x_1, x_2, x_3, x_4, x_5, x_6]$. *Similar to the previous example, build the matrix A, which shows* $\operatorname{Rank} A = 6$. *This shows that* $\frac{\partial f}{\partial x_i}$ *are linearly independent and* f *is nondegenerate. By a direct calculation, the* *general solution of* $Z(f)$ *reads* $X = \lambda_1 X_1 + \lambda_2 X_2 + \lambda_3 X_3 + \lambda_4 X_4 + \lambda_5 X_5 + \lambda_6 X_6$, *where*

$$X_1 = \begin{pmatrix} 0&0&0&0&0&0\\ 0&2&1&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0 \end{pmatrix},\ X_2 = \begin{pmatrix} 0&0&0&0&-6&-4\\ 0&0&0&0&12&8\\ 0&0&0&0&6&4\\ 0&0&0&0&6&4\\ 0&0&0&0&3&2\\ 0&0&0&0&0&0 \end{pmatrix},\ X_3 = \begin{pmatrix} 4&0&4&0&0&0\\ -8&0&-8&0&0&0\\ 5&0&5&0&0&0\\ -4&0&-4&0&0&0\\ -2&0&-2&0&0&0\\ 3&0&3&0&0&0 \end{pmatrix},$$

$$X_4 = \begin{pmatrix} -5&0&0&4&-18&0\\ 10&0&0&-8&36&0\\ 5&0&0&-4&18&0\\ -4&0&0&-13&18&0\\ -2&0&0&-2&0&0\\ 3&0&0&3&0&0 \end{pmatrix},\ X_5 = \begin{pmatrix} -5&0&0&0&-10&0\\ 10&0&0&0&20&0\\ 5&0&0&0&10&0\\ -4&0&0&0&-8&0\\ -2&0&0&0&-4&0\\ 3&0&0&0&6&0 \end{pmatrix},\ X_6 = \begin{pmatrix} 1&0&0&0&0&0\\ -2&0&0&0&0&0\\ 0&0&1&0&0&0\\ 0&0&0&1&0&0\\ 0&0&0&0&1&0\\ 0&0&0&0&0&1 \end{pmatrix}.$$

The minimal polynomials $p_i(\lambda)$ *of* $X_1, \ldots, X_6$ *are* $\lambda^2-\lambda$, $\lambda^2-1.5\lambda$, $\lambda^2-3\lambda$, $\lambda^2+3\lambda$, $\lambda^2+1.5\lambda$, *and* $\lambda^2-\lambda$, *respectively. Since* $GCD\left(p_i(\lambda), p_i'(\lambda)\right) = 1$ *for* $i = 1, \ldots, 6$, *we deduce that* f *is diagonalizable.*

4. Diagonalizing homogeneous polynomials

After determining that f is diagonalizable, Algorithm 1 iteratively computes the orthogonal idempotents $(\epsilon, I-\epsilon)$ and decomposes f until a diagonal form is obtained. The algorithm then determines whether the diagonalization is orthogonal and unitary. Our analysis shows that this process may not have polynomial-time complexity, since it involves solving a nonlinear system, which is NP-hard. However, the numerical results indicate that its computation is performed efficiently. This process is analyzed in three steps, namely compute the idempotents, diagonalize the form, and detect the orthogonality, which are discussed in Subsections 4.1–4.3, respectively. Two examples are given to demonstrate the iterative process of the diagonalization method in Subsection 4.4.

4.1. Compute idempotents

A pair of orthogonal idempotents $(\epsilon, I-\epsilon)$ is computed by imposing Rank $\epsilon = 1$ and trace$(\epsilon) = 1$ conditions on $Z(f)$. The resulting system of equations N consists of both linear and quadratic equations. Note that we have obtained the RREF C_R for $Z(f)$ in previous steps, which are transformed into $n^2 - n$ linear equations. The Rank $\epsilon = 1$ condition is reinforced by ensuring that the rows of the matrix $X = \left(x_{ij}\right)_{1\le i,j\le n}$ are linearly dependent, such that

$$\lambda_i\,(x_{11}, x_{12}, \ldots, x_{1n}) = (x_{i1}, x_{i2}, \ldots, x_{in}) \qquad \text{for } i = 2, \ldots, n.$$

They are transformed into $(n-1)^2$ quadratic equations in the form of

$$x_{1,j}x_{i+1,j+1} = x_{1,j+1}x_{i+1,j} \qquad \text{for } i, j = 1, \ldots, n-1.$$

The trace$(\epsilon) = 1$ condition is imposed directly as $x_{11} + x_{22} + \cdots + x_{nn} = 1$. The construction of these three sets of equations take $O(n^3)$, $O(n^2)$, and $O(n)$ operations, respectively. Thus, the overall construction of N requires $O(n^3)$ operations.

The system N contains $n^2 - n + 1$ linear equations and $n^2 - 2n + 1$ quadratic equations. Note that C_R is naturally sparse, and each quadratic equation comprises two terms with four variables regardless of the size of n and d. In addition, $X_1, X_2, \dots, X_n$ of $Z(f)$ often contain zero entries. Thus, even though the size of N increases quadratically as n increases, N remains relatively sparse and efficient to solve. The system N may have multiple solutions, and one will be chosen as ϵ to decompose f. If no solution is found, we deduce that f is not diagonalizable.

Nonlinear systems are often solved numerically using iterative methods like Newton's method or its variants, especially for sparse systems [19]. Solving this nonlinear system of equations is an NP-hard problem, despite its performance in the numerical experiments in Appendix A. Hence, we denote its computational complexity $O(\rho)$ in the rest of the article.

Proposition 4.1. *Computing an orthogonal idempotent ϵ has a complexity of $O(\rho)$.*

4.2. Diagonalize the form

The diagonalization algorithm decomposes f using the orthogonal idempotents $(\epsilon, I - \epsilon)$ as described in Algorithm 5. If f is not in a diagonal form after the decomposition, Algorithm 1 will continue to compute $Z(f)$ for the decomposed form of f and calculate a new ϵ. This iterative process terminates until the form of f becomes diagonal. In this section, we describe the "diagonalize the form" step in an iterative setting and denote the idempotent in iteration i as ϵ_i.

Algorithm 5 Diagonalize the form of polynomials

```
   Input: idempotent $\epsilon$
   Output: decomposed form of $f$
1: Compute the permutation matrix $P$ using a pair of orthogonal idempotents $(\epsilon, I - \epsilon)$
2: Apply the change of variables $x = Py$ to decompose $f$
3: If $f$ consists of a disjoint set of variables, it is fully diagonalized. Otherwise, replace $f$ by $g$
   with $n - 1$ variables and continue to diagonalize $f$.
```

We first calculate a pair of orthogonal idempotents $(\epsilon_1, I - \epsilon_1)$, where

$$\epsilon_1 = \begin{pmatrix} x_{11}^{(1)} & x_{12}^{(1)} & \cdots & x_{1n}^{(1)} \\ x_{21}^{(1)} & x_{22}^{(1)} & \cdots & x_{2n}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1}^{(1)} & x_{n2}^{(1)} & \cdots & x_{nn}^{(1)} \end{pmatrix} \quad \text{and} \quad I - \epsilon_1 = \begin{pmatrix} 1 - x_{11}^{(1)} & -x_{12}^{(1)} & \cdots & -x_{1n}^{(1)} \\ -x_{21}^{(1)} & 1 - x_{22}^{(1)} & \cdots & -x_{2n}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ -x_{n1}^{(1)} & -x_{n2}^{(1)} & \cdots & 1 - x_{nn}^{(1)} \end{pmatrix}.$$

The permutation matrix P_1 of f now takes the form of a combination of ϵ and $I - \epsilon$ as explained in Proposition 3.7 in [9], where

$$P_1 = \begin{pmatrix} x_{11}^{(1)} & x_{12}^{(1)} & \cdots & x_{1r}^{(1)} \\ -x_{21}^{(1)} & 1 - x_{22}^{(1)} & \cdots & -x_{2n}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ -x_{n1}^{(1)} & -x_{n2}^{(1)} & \cdots & 1 - x_{nn}^{(1)} \end{pmatrix}.$$

Decompose f by applying a change of variables

$$\left(y_1^{(1)}, y_2^{(1)}, \dots, y_n^{(1)}\right)^T = P_1 \left(y_1, y_2, \dots, y_n\right)^T ,$$

where $y_1^{(1)}, y_2^{(1)}, \ldots, y_n^{(1)}$ are new variables in the first iteration. We then obtain

$$f = c_1 \left(y_1^{(1)}\right)^d + g_1 \left(y_2^{(1)}, y_3^{(1)}, \ldots, y_n^{(1)}\right),$$

where c_1 is a d-th power linear form and g_1 is a nondegenerate form of degree d in $n-1$ variables.

If g_1 is diagonal, the diagonalization process completes, and we continue to determine the properties of the diagonalization; otherwise, we return to Step 5 of Algorithm 1. After computing the new center $Z(g_1)$ and the corresponding orthogonal idempotent $(\epsilon_2, I - \epsilon_2)$, we have

$$P_2 = \begin{pmatrix} I_1 & 0 & 0 & \cdots & 0 \\ 0 & x_{22}^{(2)} & x_{23}^{(2)} & \cdots & x_{2r}^{(2)} \\ 0 & -x_{32}^{(2)} & 1 - xp_{33}^{(2)} & \cdots & -x_{3n}^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & -x_{n2}^{(2)} & -x_{n3}^{(2)} & \cdots & 1 - x_{nn}^{(2)} \end{pmatrix}.$$

Then we decompose g_1 by applying the change of variables

$$\left(y_2^{(2)}, \ldots, y_n^{(2)}\right)^T = P_2 \left(y_2^{(1)}, \ldots, y_n^{(1)}\right)^T$$

and we have

$$g_1 = c_2 \left(y_2^{(2)}\right)^d + g_2 \left(y_3^{(2)}, \ldots, y_n^{(2)}\right).$$

Here, g_2 is a nondegenerate form of degree d in $n-2$ variables and is independent from $y_1^{(1)}$ and $y_2^{(2)}$.

Let $\left(y_1^{(k)}, y_2^{(k)}, \ldots, y_n^{(k)}\right)^T = P_k \left(y_1^{(k-1)}, y_2^{(k-1)}, \ldots, y_n^{(k-1)}\right)^T$, where the k-th iteration of the permutation matrix P_k is defined as

$$P_k = \left(\begin{array}{c|cccc} I_k & 0 & 0 & \cdots & 0 \\ \hline 0 & x_{k,k}^{(k)} & x_{k,k+1}^{(k)} & \cdots & x_{k,n}^{(k)} \\ 0 & -x_{k+1,k}^{(k)} & 1 - x_{k+1,k+1}^{(k)} & \cdots & -x_{k+1,n}^{(k)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & -x_{n,k}^{(k)} & -x_{n,n+1}^{(k)} & \cdots & 1 - x_{n,n}^{(k)} \end{array}\right).$$

This process continues until the form g_k becomes diagonal. The center then takes the form of

$$Z(f) = Z\left(c_1 (y_1^{(k)})^d\right) \times Z\left(c_2 (y_2^{(k)})^d\right) \times \cdots \times Z\left(c_n (y_n^{(k)})^d\right),$$

and f is transformed into a sum of n d-th powers by the linear transformation

$$f = c_1 \left(y_1^{(k)}\right)^d + c_2 \left(y_2^{(k)}\right)^d + \cdots + c_n \left(y_n^{(k)}\right)^d.$$

Let $P = P_{n-1} P_{n-2} \cdots P_1$. We then have

$$(y_1, y_2, \ldots, y_n)^T = P^{-1} \left(y_1^{(k)}, y_2^{(k)}, \ldots, y_n^{(k)}\right)^T = P^{-1} (z_1, z_2, \ldots, z_n)^T,$$

where $y_i^{(k)} = z_i$. Now f is represented as

$$f(P^{-1} z) = c_1 z_1^d + c_2 z_2^d + \cdots + c_n z_n^d.$$

If the original form of f is degenerate, a matrix $\bar{P}$ is built as follows:

$$\bar{P} = \left(\begin{array}{c|c} P & 0 \\ \hline 0 & I_{\bar{n}-n} \end{array}\right).$$

There is a linear transformation $(z_1, z_2, \dots, z_{\bar{n}})^T = \bar{P}A^{-1}(x_1, x_2, \dots, x_{\bar{n}})^T$ between the original variable x_i and z_i. Let $Q = \bar{P}A^{-1} = \left(q_{ij}\right)_{1\le i,j\le \bar{n}}$. We then have $z_i = \sum_{j=1}^{\bar{n}} q_{ij}x_j$. Substitute this back to the sum of powers and we get

$$f = c_1\left(\sum_{i=1}^{\bar{n}} q_{1i}x_i\right)^d + \cdots + c_{\bar{n}}\left(\sum_{i=1}^{\bar{n}} q_{\bar{n}i}x_i\right)^d.$$

The diagonal form step calculates P and its inverse, which take $O(n^2)$ and $O(n^3)$ operations, respectively. While the change of variables takes at most $O(n^2t)$ operations, we need to expand t terms of f, which involves several multivariate polynomial multiplications. Since multiplying two polynomials takes $O(n \log n \log\log n)$ operations for polynomials with $d < n$, expanding each term takes at most $O(nd \log n \log\log n)$ operations [16]. Therefore, the overall complexity of this step is $O(ndt \log n \log\log n)$.

Proposition 4.2. *Diagonalizing the form of f takes $O(ndt \log n \log\log n)$ arithmetic operations.*

4.3. Detect orthogonality

We determine the orthogonality and unitarity of the diagonalization using the permutation matrix P computed from the previous step. If PP^T is diagonal, the diagonalization is orthogonal. If $P\bar{P}^T$ is diagonal, the diagonalization is unitary. This process involves the calculation of a transpose P^T and a conjugate transpose $\bar{P}^T$ of matrix P, which both take $O(n^2)$ arithmetic operations. In addition to two matrix multiplications with a complexity of $O(n^3)$, the overall computational complexity of this step is $O(n^3)$.

Proposition 4.3. *Determining orthogonality and unitarity of f takes $O(n^3)$ arithmetic operations.*

In summary, the iterative diagonalization process consists of computing the center, computing the idempotents, and diagonalizing form, which have a complexity of $O(n^3d^3t + n^6)$, $O(\rho)$, and $O(ndt \log n \log\log n)$, respectively. This iterative process lasts at most $n - 1$ iterations, and n is reduced by 1 in each iteration. Therefore, the entire process takes $O(\log n)$ at the cost of of the three steps above. Lastly, the orthogonality and unitarity are determined straightforwardly with $O(n^3)$ operations, which does not affect the overall complexity.

4.4. Some examples

Example 4.4. *Consider the quadruple quintic polynomial f in Example 3.4, whose diagonalizability has been verified. Build and solve the system N using its center $Z(f)$ with conditions of Rank 1 and trace 1, which may give more than one solution. Choose one solution as ϵ_1 with a nonzero 1,1-th entry and obtain a pair of orthogonal idempotents $(\epsilon_1, I_4 - \epsilon_1)$, where*

$$\epsilon_1 = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -1 \\ \frac{3}{2} & \frac{3}{2} & -\frac{3}{2} & -3 \\ 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -1 \end{pmatrix}, \quad I_4 - \epsilon_1 = \begin{pmatrix} \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & 1 \\ -\frac{3}{2} & -\frac{1}{2} & \frac{3}{2} & 3 \\ 0 & 0 & 1 & 0 \\ -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -2 \end{pmatrix}.$$

According to the pair $(\epsilon_1, I_4 - \epsilon_1)$, we take a change of variables $y = P_1 x$, where

$$P_1 = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -1 \\ -\frac{3}{2} & -\frac{1}{2} & \frac{3}{2} & 3 \\ 0 & 0 & 1 & 0 \\ -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & 2 \end{pmatrix}.$$

By direct computation, we have

$$\begin{aligned} f(P_1^{-1}y) = {} & 1024y_1^5 + 7807y_2^5 - 51915y_2^4y_3 - 64955y_2^4y_4 + 138310y_2^3y_3^2 + 345900y_2^3y_3y_4 + 216310y_2^3y_4^2 \\ & - 184350y_2^2y_3^3 - 691410y_2^2y_3^2y_4 - 864450y_2^2y_3y_4^2 - 360310y_2^2y_4^3 + 122885y_2y_3^4 + 614460y_2y_3^3y_4 \\ & + 1152210y_2y_3^2y_4^2 + 960300y_2y_3y_4^3 + 300155y_2y_4^4 - 32768y_3^5 - 204805y_3^4y_4 - 512030y_3^3y_4^2 \\ & - 640070y_3^2y_4^3 - 400075y_3y_4^4 - 100031y_4^5. \end{aligned}$$

Note that the decomposition takes the form of $f(P_1^{-1}y) = t(y_1) + g(y_2, y_3, y_4)$, where $t(y_1) = -1024y_1^5$. Since g is not diagonal, we continue to diagonalize g. Calculate $Z(g) = \{X_g \in \mathbb{C}^{3\times 3} | (H_gX_g)^T = X_gH_g\}$, and the general solution is

$$X_g = \lambda_1 \begin{pmatrix} -8 & 8 & 13 \\ -16 & 11 & 16 \\ 1 & 0 & 0 \end{pmatrix} + \lambda_2 \begin{pmatrix} -16 & 11 & 16 \\ -12 & 2 & 12 \\ 0 & 1 & 0 \end{pmatrix} + \lambda_3 \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

We obtain a new pair of orthogonal idempotents $(\epsilon_2, I_3 - \epsilon_2)$ based on X_g, where

$$\epsilon_2 = \begin{pmatrix} \frac{3}{2} & -\frac{3}{2} & -\frac{3}{2} \\ -2 & 2 & 2 \\ \frac{5}{2} & -\frac{5}{2} & -\frac{5}{2} \end{pmatrix}, \quad I_3 - \epsilon_2 = \begin{pmatrix} -\frac{1}{2} & \frac{3}{2} & \frac{3}{2} \\ 2 & -1 & -2 \\ -\frac{5}{2} & \frac{5}{2} & \frac{7}{2} \end{pmatrix}.$$

We then take a change of variables $z = P_2 y$ according to $(\epsilon_2, I_3 - \epsilon_2)$, where

$$P_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{3}{2} & -\frac{3}{2} & -\frac{3}{2} \\ 0 & 2 & -1 & -2 \\ 0 & -\frac{5}{2} & \frac{5}{2} & \frac{7}{2} \end{pmatrix}.$$

By direct computation, we have

$$g(P_2^{-1}z) = -0.13162z_2^5 - 31z_3^5 - 320z_3^4z_4 - 1280z_3^3z_4^2 - 2560z_3^2z_4^3 - 2560z_3z_4^4 - 1024z_4^5.$$

Note that the decomposition takes the form of $g(P_2^{-1}z) = t(z_2) + h(z_3, z_4)$, where $t(z_2) = -0.13162z_2^5$. Since h is not diagonal, we continue to diagonalize h. Compute $Z(h) = \{X_h \in \mathbb{C}^{2\times 2} | (H_hX_h)^T = X_hH_h\}$, and the general solution is

$$X_h = \lambda_1 \begin{pmatrix} -2 & 0 \\ 1 & 0 \end{pmatrix} + \lambda_2 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Compute the orthogonal idempotents $(\epsilon_3, I_2 - \epsilon_3)$. We then have

$$\epsilon_3 = \begin{pmatrix} 1 & 0 \\ -\frac{1}{2} & 0 \end{pmatrix}, \quad I_2 - \epsilon_3 = \begin{pmatrix} 0 & 0 \\ \frac{1}{2} & 1 \end{pmatrix}.$$

Take a change of variables $w = P_3z$ according to the pair $(\epsilon_3, I_2 - \epsilon_3)$, where

$$P_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{2} & 1 \end{pmatrix}.$$

By direct computation, we have the following decomposition:

$$h(P_3^{-1}w) = w_3^5 - 1024w_4^5.$$

Since h is in diagonal form, the diagonalization process terminates. The final change of variables $w = Px$ takes the form of

$$P = P_3P_2P_1 = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -1 \\ -\frac{3}{2} & 0 & 0 & \frac{3}{2} \\ -2 & 0 & 1 & 2 \\ 1 & -\frac{1}{2} & 1 & \frac{1}{2} \end{pmatrix}$$

and we obtain

$$f(P^{-1}w) = 1024w_1^5 - 0.13162w_2^5 + w_3^5 - 1024w_4^5.$$

Finally, f is diagonalized as

$$f(x_1, x_2, x_3, x_4) = (2x_1 + 2x_2 - 2x_3 - 4x_4)^5 + (x_1 - x_4)^5 + (-2x_1 + x_3 + 2x_4)^5 + (-4x_1 + 2x_2 - 4x_3 + 2x_4)^5 .$$

In addition, after examining the permutation matrix P, we deduce that the diagonalization of f is neither orthogonal nor unitary.

Example 4.5. *Consider the sextuple septic polynomial f defined in Example 3.5, whose diagonalizability has been verified. Build and solve the system N based on Z(f), we calculate a pair of orthogonal idempotents $(\epsilon_1, I_6 - \epsilon_1)$, where*

$$\epsilon_1 = \begin{pmatrix} \frac{4}{9} & 0 & \frac{4}{9} & 0 & 0 & 0 \\ -\frac{8}{9} & 0 & -\frac{8}{9} & 0 & 0 & 0 \\ \frac{5}{9} & 0 & \frac{5}{9} & 0 & 0 & 0 \\ -\frac{4}{9} & 0 & -\frac{4}{9} & 0 & 0 & 0 \\ -\frac{2}{9} & 0 & -\frac{2}{9} & 0 & 0 & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & 0 \end{pmatrix}, \quad I_6 - \epsilon_1 = \begin{pmatrix} \frac{5}{9} & 0 & -\frac{4}{9} & 0 & 0 & 0 \\ \frac{8}{9} & 1 & \frac{8}{9} & 0 & 0 & 0 \\ -\frac{5}{9} & 0 & \frac{4}{9} & 0 & 0 & 0 \\ \frac{4}{9} & 0 & \frac{4}{9} & 1 & 0 & 0 \\ \frac{2}{9} & 0 & \frac{2}{9} & 0 & 1 & 0 \\ -\frac{1}{3} & 0 & -\frac{1}{3} & 0 & 0 & 1 \end{pmatrix}.$$

According to $(\epsilon_1, I_6 - \epsilon_1)$, we take a change of variables $y = P_1x$, where

$$P_1 = \begin{pmatrix} \frac{4}{9} & 0 & \frac{4}{9} & 0 & 0 & 0 \\ \frac{8}{9} & 1 & \frac{8}{9} & 0 & 0 & 0 \\ -\frac{5}{9} & 0 & \frac{4}{9} & 0 & 0 & 0 \\ \frac{4}{9} & 0 & \frac{4}{9} & 1 & 0 & 0 \\ \frac{2}{9} & 0 & \frac{2}{9} & 0 & 1 & 0 \\ -\frac{1}{3} & 0 & -\frac{1}{3} & 0 & 0 & 1 \end{pmatrix}$$

and

$$\begin{aligned}
f(P_1^{-1}y) = {} & -291.9293y_1^7 + 1y_2^7 - 14y_2^6y_3 + 84y_2^5y_3^2 - 280y_2^4y_3^3 + 560y_2^3y_3^4 - 672y_2^2y_3^5 + 448y_2y_3^6 - 128y_3^7 \\
& - 7y_3^6y_4 + 14y_3^6y_5 - 21y_3^6y_6 + 21y_3^5y_4^2 + 126y_3^5y_4y_6 - 84y_3^5y_5^2 + 189y_3^5y_6^2 - 35y_3^4y_4^3 - 315y_3^4y_4^2y_6 \\
& - 945y_3^4y_4y_6^2 + 280y_3^4y_5^3 - 945y_3^4y_6^3 + 35y_3^3y_4^4 + 420y_3^3y_4^3y_6 + 1890y_3^3y_4^2y_6^2 + 3780y_3^3y_4y_6^3 \\
& - 560y_3^3y_5^4 + 2835y_3^3y_6^4 - 21y_3^2y_4^5 - 315y_3^2y_4^4y_6 - 1890y_3^2y_4^3y_6^2 - 5670y_3^2y_4^2y_6^3 - 8505y_3^2y_4y_6^4 + 672y_3^2y_5^5 \\
& - 5103y_3^2y_6^5 + 7y_3y_4^6 + 126y_3y_4^5y_6 + 945y_3y_4^4y_6^2 + 3780y_3y_4^3y_6^3 + 8505y_3y_4^2y_6^4 + 10206y_3y_4y_6^5 \\
& - 448y_3y_5^6 + 5103y_3y_6^6 - 2y_4^7 + 14y_4^6y_5 - 21y_4^6y_6 - 84y_4^5y_5^2 - 189y_4^5y_6^2 + 280y_4^4y_5^3 - 945y_4^4y_6^3 \\
& - 560y_4^3y_5^4 - 2835y_4^3y_6^4 + 672y_4^2y_5^5 - 5103y_4^2y_6^5 - 448y_4y_5^6 - 5103y_4y_6^6 + 2443y_5^7 + 10206y_5^6y_6 \\
& + 20412y_5^5y_6^2 + 22680y_5^4y_6^3 + 15120y_5^3y_6^4 + 6048y_5^2y_6^5 + 1344y_5y_6^6 - 2059y_6^7.
\end{aligned}$$

Note that the decomposition takes the form of $f(P_1^{-1}y) = t(y_1) + g(y_2, \dots, y_6)$, where $t(y_1) = -291.9293y_1^7$. Since g is not diagonal, we continue to diagonalize g. Repeat the process above for another four iterations and we obtain the following P_i for the changes of variables $y^{(i+1)} = P_iy^{(i)}$ for $i = 2, \dots, 5$, where

$$P_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad P_3 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{4}{9} & -\frac{4}{9} & 0 & -\frac{4}{3} \\ 0 & 0 & -\frac{4}{9} & \frac{13}{9} & 0 & \frac{4}{3} \\ 0 & 0 & -\frac{2}{9} & \frac{2}{9} & 0 & \frac{2}{3} \\ 0 & 0 & \frac{1}{3} & -\frac{1}{3} & 0 & 0 \end{pmatrix},$$

$$P_4 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{4}{9} & \frac{8}{9} & \frac{4}{3} \\ 0 & 0 & 0 & \frac{2}{9} & \frac{5}{9} & \frac{2}{3} \\ 0 & 0 & 0 & -\frac{1}{3} & \frac{2}{3} & 0 \end{pmatrix}, \quad P_5 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \frac{2}{3} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

The final change of variables $z = y^{(5)} = Px$ then takes the form of

$$P = P_5P_4P_3P_2P_1 = \begin{pmatrix} \frac{4}{9} & 0 & \frac{4}{9} & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{4}{9} & -\frac{4}{9} & 0 & -\frac{4}{3} \\ \frac{4}{9} & 0 & 0 & 0 & \frac{8}{9} & 0 \\ 0 & 0 & 0 & 0 & 1 & \frac{2}{3} \\ 0 & 0 & 0 & -\frac{1}{3} & \frac{2}{3} & 0 \end{pmatrix}$$

and we obtain

$$f(P^{-1}z) = -291.929z_1^7 + z_2^7 + 291.929z_3^7 + 291.929z_4^7 + 2187z_5^7 + 2187z_6^7.$$

Lastly, f is diagonalized as

$$f(x_1, \dots, x_6) = (-x_1 - x_3)^7 + (2x_1 + x_2)^7 + (x_3 - x_4 - 3x_6)^7 + (x_1 + 2x_5)^7 + (3x_5 + 2x_6)^7 + (-x_4 + 2x_5)^7.$$

The examination of P shows that this diagonalization is neither orthogonal nor unitary.

5. Comparison with other diagonalization techniques

The current diagonalization techniques, including [1–4], have successfully provided criteria and algorithms for diagonalizing homogeneous polynomials. While they have made significant progress in the field, they are either only applicable to specific types of polynomials or appeal to highly nonlinear tasks such as polynomial factorization. We provide more discussion of their complexity below.

The seminal work of Kayal [1] provides a criterion and diagonalization algorithm for f with an arbitrary n and d as described in Algorithm 6. By reducing the problem to polynomial factorizations, this technique achieves randomized polynomial time complexity for real polynomials. However, this complexity result does not hold for complex cases. In Step 1, the determinant $\mathrm{DET}(H)$ may be factorized using Kaltofen's algorithm [11] or the black box factorization algorithm of Kaltofen and Trager [20], which both admit randomized polynomial time. However, Kaltofen's algorithm uses Hensel lifting for d iterations, which leads to an exponential blowup of sizes in modules where the previous computations cannot be reused [21]. While more work has been done to improve its efficiency for high-degree polynomials with a sparse representation, the complexity is still quite high for a large d. In Step 2, the coefficients a_i can be obtained efficiently by solving a dense linear algebra problem. Similarly, the complexity of the root findings in Step 3 is polynomial and the equivalence problem is solved in randomized polynomial time.

Algorithm 6 Kayal's algorithm

```
1: Compute the Hessian determinant $\mathrm{DET}(H)$ of $f$ and check if it is identically 0 and can be factorized as $\mathrm{DET}(H) = c\prod_{i=1}^{n} l_i(x_1, x_2, \ldots, x_n)^{d-2}$ where $l_i$ are linear forms and $c \in \mathbb{C}$. Reject the results if it is not possible.
2: Calculate the constants $a_i \in \mathbb{C}$ such that $f(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} \lambda_i l_i(x_1, x_2, \ldots, x_n)^d$. Terminate if it is not possible.
3: Check if all $a_i$ values have their $d$-th roots in $\mathbb{C}$. Reject them if it is not possible.
```

Koiran [2] proposed a deterministic criterion to determine the diagonalizability for polynomials f with degree $d = 3$, as described in Algorithm 7. In contrast to Kayal's technique, it does not factorize Hessian determinant $\mathrm{DET}(H)$ explicitly. Recall that a homogeneous polynomial f with $d = 3$ can be associated with a symmetric tensor T of order 3, where $f(x_1, x_2, \ldots, x_n) = \sum_{i,j,k=1}^{n} T_{ijk}x_ix_jx_k$. We have $\frac{\partial^3 f}{\partial x_i \partial x_j \partial x_k} = 6T_{ijk}$, and the i-th slice of T is the symmetric matrix T_i with the entries $(T_i)_{jk} = T_{ijk}$.

Algorithm 7 Koiran's algorithm

```
1: On input $f \in \mathbb{C}[x_1, x_2, \ldots, x_n]$, pick a random matrix $R \in M_n(\mathbb{C})$ and set $h(x) = f(Rx)$.
2: Let $T_1, T_2, \ldots, T_n$ of $h$ be the slices of $h$. If $T_1$ is singular, reject. Otherwise, compute $T_1' = T_1^{-1}$.
3: If the matrices $T_1'T_k$ commute and are all diagonalizable over $\mathbb{C}$, accept them. Otherwise, reject them.
```

Koiran's criterion iterates through n slices of T_i to determine their diagonalizability. Instead of factorizing explicitly, Koiran used the existence of the factorization deterministically to find a point where H does not vanish. It includes routines such as calculation of the roots from polynomials, which can be obtained through Hurwitz determinants. Koiran and Saha [3] provided a further improvement and a complexity analysis of this algorithm, which showed that it requires $O(n^{\omega+2})$ arithmetic operations over the complex field and that ω is a feasible exponent for matrix multiplication. Koiran and Saha also presented a new randomized black box criterion that uses the extraction of complex polynomial roots to determine the diagonalizability of f with $d > 3$. This algorithm takes $O(n^2 d \log^2 d \log\log d + n^{\omega+1})$ arithmetic operations for $f \in \mathbb{C}$.

Robeva [4] studied the diagonalization from the perspective of symmetric tensors and proposed to use the tensor power method from Anandkumar et al. [22] or the tensor decomposition technique from Brachat et al. [23], which are based on the properties of Hankel matrices. However, the algorithm is difficult to construct and requires a large nonlinear system of equations to be solved.

Some other studies focused on determining the properties of diagonalization, such as orthogonality and unitarity [4–7]. These properties are useful for areas like theoretical computer science and scientific computing. Kolda [6] studied the orthogonal decomposition of tensors and computed the best Rank-1 approximations by solving a minimization problem with orthogonality constraints. Kolda used an alternating least squares approach to solve the problem, which converges slowly and becomes a computationally challenging task. Robeva [4] tackled the issue of the orthogonal decompositions of symmetric tensors using the tensor power method. Boralevi et al. [7] proposed a randomized algorithm to check the orthogonality and unitarity of a tensor and decompose it if possible, which is based on singular value decompositions. Koiran [5] expanded the study for orthogonal and unitary decompositions over the field of complex numbers. Their techniques need to solve a large system of quadratic equations. In contrast, Huang et al.'s diagonalization algorithm only requires basic matrix inversions and multiplications to determine the orthogonality and unitarity of the diagonalization using the computed permutation matrix P.

To conclude, many techniques have been proposed as criteria for diagonalizability and diagonalization, or criteria for orthogonality and unitarity. They either only apply to limited polynomials or use computationally expensive techniques. While some researchers provided a complexity analysis, few gave sufficient numerical results for illustration and comparison. In contrast, our complexity analysis is validated by numerical experiments with concrete examples. This makes the center-based diagonalization algorithm a competitive alternative in this field.

6. Summary

In this article, we provide detailed formulations and a complexity analysis of the diagonalization technique based on Harrison's center theory. It consists of a criterion for diagonalizability and a diagonalization algorithm. The technique is divided into six steps, and their computational complexity is summarized in Table 1. This table also contains the convergence rates from the numerical results given in Appendix A. The convergence is measured for f with various n and d values, which reflect the given problem sizes.

Among the six steps, the steps of detecting nondegeneracy, computing the center, and diagonalizing the form have the highest theoretical complexity. Their costs depend on the number of terms t in f, which increases if either n or d increases. The steps of detecting nondegeneracy and computing the center need to compute multiple first- or second-order derivatives of f. Diagonalizing the form applies the change of variables to f and requires operations like polynomial multiplications or power expansions. Their complexity is consistent with their high convergence rates for n and d, as shown in the numerical results.

Table 1. Complexity of each step of diagonalization.

Step	Complexity	Convergence n	Convergence d
Detect nondegeneracy	$O(n^2t)$	$O(n^{4.02})$	$O(d^{2.50})$
Compute center	$O(n^3d^3t + n^6)$	$O(n^{6.07})$	$O(d^{2.85})$
Detect diagonalizability	$O(n^5)$	$O(n^{1.31})$	$O(d^{-0.020})$
Compute idempotent	$O(\rho)$	$O(n^{2.5})$	$O(d^{0.004})$
Diagonalize form	$O(ndt \log n \log\log n)$	$O(n^{5.12})$	$O(d^{2.93})$
Detect orthogonality	$O(n^3)$	$O(n^{0.062})$	$O(d^{-0.40})$
Total		$O(n^{4.26})$	$O(d^{2.39})$

The complexity of the other three steps is independent of t and shows markedly lower convergence rates for n and d. While we solve a quadratic system of equations N to compute the idempotents, the system's dimension only depends on n, and its entries are relatively sparse. Consequently, the convergence of its computational costs for n is below quadratic and remains relatively unchanged when d increases. Similarly, the increase in d does not affect the costs of detecting diagonalizability.

The criterion of diagonalizability consists of detecting nondegeneracy, computing the center, and detecting diagonalizability, and the complexity of this process is

$$O(n^2t) + O(n^3d^3t + n^6) + O(n^5) = O(n^3d^3t + n^6).$$

Since t grows polynomially if either n or d increases while the other is fixed, we deduce the complexity of this criterion is polynomially bounded for n and d. The diagonalization process comprises an iterative process, including steps of computing the center, computing the idempotent, and diagonalizing the form, and terminates with at most $n-1$ steps. Consequently, its complexity is

$$O(\log n)\left(O(n^3d^3t + n^6) + O(\rho) + O(ndt \log n \log\log n)\right).$$

Since solving the system N is NP-hard, the complexity of this process is not polynomial.

We compared our technique with other criteria and diagonalization techniques regarding their computational costs. Many of them lack a detailed complexity analysis and rely on methods such as the polynomial factorization and the tensor power method, which require randomized polynomial time. Some other techniques like [1–3] may fail with a small probability. In contrast, the center-based diagonalization technique only requires solving sparse linear and quadratic systems and is applicable for f with an arbitrary n and d.

The exact complexity of solving its quadratic systems of equations is still unknown and needs to be addressed in future studies. Additionally, the numerical experiments showed that the rounding errors accumulated during the iterative process may affect the accuracy of the solution. Future studies should further investigate the algorithm's stability and error bounds, which are crucial to the practicability of this technique. Moreover, current research mostly focuses on one polynomial. However, real-world applications may demand simultaneous criteria and diagonalization of more than one polynomial. We also want to extend the complexity analysis to the direct sum decomposition and Waring decomposition of the polynomials.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

L. Fang was partially supported by the Natural Science Foundation of Xiamen (Grant No. 3502Z202371014). H. L. Huang was partially supported by the Key Program of Natural Science Foundation of Fujian Province (Grant No. 2024J02018) and the National Natural Science Foundation of China (Grant No. 12371037).

Conflict of interest

The authors declare there are no conflicts of interest.

References

1. N. Kayal, Efficient algorithms for some special cases of the polynomial equivalence problem, in *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, **1** (2011), 1409–1421. https://doi.org/10.5555/2133036.2133144
2. P. Koiran, M. Skomra, Derandomization and absolute reconstruction for sums of powers of linear forms, *Theor. Comput. Sci.*, **887** (2021), 63–84. https://doi.org/10.1016/j.tcs.2021.07.005
3. P. Koiran, S. Saha, Absolute reconstruction for sums of powers of linear forms: degree 3 and beyond, *Comput. Complexity*, **32** (2023), 1–66. https://doi.org/10.1007/s00037-023-00239-8
4. E. Robeva, Orthogonal decomposition of symmetric tensors, *SIAM J. Matrix Anal. Appl.*, **37** (2016), 86–102. https://doi.org/10.1137/140989340
5. P. Koiran, Orthogonal tensor decomposition and orbit closures from a linear algebraic perspective, *Linear Multilinear Algebra*, **69** (2021), 2353–2388. https://doi.org/10.1080/03081087.2019.1674771
6. T. Kolda, Orthogonal tensor decompositions, *SIAM J. Matrix Anal. Appl.*, **23** (2000), 243–255. https://doi.org/10.1137/S0895479800368354
7. A. Boralevi, J. Draisma, E. Horobeţ, E. Robeva, Orthogonal and unitary tensor decomposition from an algebraic perspective, *Isr. J. Math.*, **222** (2017), 223–260. https://doi.org/10.1007/s11856-017-1588-6
8. D. K. Harrison, A grothendieck ring of higher degree forms, *J. Algebra*, **35** (1975), 123–138. https://doi.org/10.1016/0021-8693(75)90039-3
9. H. L. Huang, H. Lu, Y. Ye, C. Zhang, Diagonalizable higher degree forms and symmetric tensors, *Linear Algebra Appl.*, **613** (2021), 151–169. https://doi.org/10.1016/j.laa.2020.12.018
10. H. L. Huang, H. Lu, Y. Ye, C. Zhang, On centres and direct sum decompositions of higher degree forms, *Linear Multilinear Algebra*, **70** (2022), 7290–7306. https://doi.org/10.1080/03081087.2021.1985057

11. E. Kaltofen, Factorization of polynomials given by straight-line programs, *Adv. Comput. Res.*, **5** (1989), 375–412.

12. H. L. Huang, L. Liao, H. Lu, Y. Ye, C. Zhang, Harrison center and products of sums of powers, *Commun. Math. Stat.*, **2023** (2023). https://doi.org/10.1007/s40304-023-00367-1

13. H. L. Huang, H. Lu, Y. Ye, C. Zhang, Centers of multilinear forms and applications, *Linear Algebra Appl.*, **673** (2023), 160–176. https://doi.org/10.1016/j.laa.2023.05.012

14. G. H. Golub, C. F. Van Loan, *Matrix Computations*, JHU Press, Baltimore, 2013.

15. H. Y. Cheung, T. C. Kwok, L. C. Lau, Fast matrix rank algorithms and applications, *J. ACM*, **60** (2013), 1–25. https://doi.org/10.1145/2528404

16. D. G. Cantor, E. Kaltofen, On fast multiplication of polynomials over arbitrary algebras, *Acta Inf.*, **28** (1991), 693–701. https://doi.org/10.1007/BF01178683

17. S. Boyd, L. Vandenberghe, *Introduction to Applied Linear Algebra: Vectors, Matrices, and Least Squares*, Cambridge University Press, Cambridge, 2018. https://doi.org/10.1017/9781108583664

18. D. Bini, V. Y. Pan, *Polynomial and Matrix Computations: Fundamental Algorithms*, Springer Science & Business Media, Basel, 2012. https://doi.org/10.1007/978-1-4612-0265-3

19. A. Cordero, J. L. Hueso, E. Martínez, J. R. Torregrosa, Increasing the convergence order of an iterative method for nonlinear systems, *Appl. Math. Lett.*, **25** (2012), 2369–2374. https://doi.org/10.1016/j.aml.2012.07.005

20. E. Kaltofen, B. M. Trager, Computing with polynomials given by black boxes for their evaluations: Greatest common divisors, factorization, separation of numerators and denominators, *J. Symb. Comput.*, **9** (1990), 301–320. https://doi.org/10.1016/S0747-7171(08)80015-6

21. A. Sinhababu, T. Thierauf, Factorization of polynomials given by arithmetic branching programs, *Comput. Complexity*, **30** (2021). https://doi.org/10.1007/s00037-021-00215-0

22. A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, M. Telgarsky, Tensor decompositions for learning latent variable models, *J. Mach. Learn. Res.*, **15** (2014) 2773–283. https://doi.org/10.5555/2627435.2697055

23. J. Brachat, P. Comon, B. Mourrain, E. Tsigaridas, Symmetric tensor decopmosition, *Linear Algebra Appl.*, **433** (2010), 1851–1872. https://doi.org/10.1016/j.laa.2010.06.046

24. Y. Chen, E. J. Candès, Solving random quadratic systems of equations is nearly as easy as solving linear systems, *Commun. Pure Appl. Math.*, **70** (2017), 822–883. https://doi.org/10.1002/cpa.21638

Appendix

A. Computational cost of diagonalization

In this section, we provide numerical results to validate the impacts of n and d on the costs of the diagonalization technique. All the numerical experiments are conducted in MATLAB R2022a on a personal computer with a Microsoft Windows 10 operating system and an Intel® Core™ i9-10900

CPU @ 2.80 GHz configuration with 64.0 GB of RAM. The implementations were developed by Fang and are available from Bitbucket*. The computational costs for various dimensions n and degrees d are given in Examples A.1 and A.2, respectively.

Example A.1. *Consider a homogeneous polynomial*

$$f = \sum_{i=1}^{n} (\lambda_{i1}x_1 + \cdots + \lambda_{in}x_n)^5 ,$$

where λ_{ij} are randomly generated such that $\{\lambda_{i1}, \lambda_{i2}, \ldots, \lambda_{in}\}$ are linearly independent. A numerical experiment was conducted to compare the computational costs for diagonalizing f with $n = 2, \ldots, 7$, and the statistics are shown in Table A1. It includes the execution time of the six major steps of Algorithm 1 and the entire program for different n values. The execution time was estimated as the average elapsed time of 10 executions of the diagonalization program, measured in seconds. Table A1 also includes their convergence rates to verify the previous complexity analysis. Note that the steps of computing the center, computing the idempotent, and diagonalizing the form may be executed several times during the iterative process, and Table A1 only shows the execution time in the first iteration.

Table A1 shows that the total execution time of the diagonalization increases in the order of $O(n^{4.26})$, which is markedly lower than the theoretical upper bounds given in Section 6. Among the six steps, the step of computing the center has the highest convergence rate of $O(n^{6.07})$. While the step of diagonalizing the form has a lower convergence of $O(n^{5.12})$, its execution time in the first iteration takes up over 38% of the total execution time for $n = 7$. Since both of these steps involve iterating through t terms of f, its costs increase significantly if either n or d increases.

While the step of detecting nondegeneracy has a high convergence rate of $O(n^{4.02})$, it only takes 0.14 seconds for $n = 7$ and is trivial within the algorithm. The other steps, like detecting diagonalizability, computing the idempotent, and detecting orthogonality, all have low convergence compared with their upper bounds. Since the step of detecting orthogonality only involves a single matrix inversion and multiplication, it has a low impact on the program execution time for all n and shows near-zero convergence.

The computational costs for solving the linear system $CX_v = 0$ and the quadratic system N are important components of the complexity analysis, as they are not easily parallelizable. We provide additional execution time and the corresponding convergence rates for the matrix C and the system N in Table A2, including the solving time, dimension, and sparsity. The dimension refers to the number of equations in the system, and the sparsity is the ratio of nonzero entries. Table A2 shows that the costs for computing the RREF for the matrix C increase drastically in the order of $O(n^{6.66})$. This is reasonable, as C's dimension increases with the convergence $O(n^{4.50})$, and the sparsity only decreases in the order of $O(n^{-1.08})$. In contrast, the costs for solving the system N increase in the order of $O(n^{1.85})$. While the dimension of the system N increases quadratically, as shown in Section 4.1, its quadratic equations only involve four variables and were efficiently solved in the experiments.

Example A.2. *Consider a homogeneous polynomial*

$$f = (x_1 + 2x_2 - x_3)^d + (2x_1 + 2x_2 - x_3)^d + (x_1 - 2x_2 + 2x_3)^d ,$$

*https://bitbucket.org/fanglishan/diagonalization/src/master/

Table A1. Execution time (in seconds) of diagonalization with various values of n.

Step \ n	2	3	4	5	6	7	**Convergence**
Detect nondegeneracy	9.00×10^{-4}	1.40×10^{-3}	4.90×10^{-3}	0.015	0.037	0.14	4.02
Compute the center	1.10×10^{-3}	4.50×10^{-3}	0.031	0.13	0.59	1.92	6.07
Detect diagonalizability	0.022	0.036	0.046	0.056	0.082	0.13	1.31
Detect idempotent	0.099	0.19	0.39	0.72	1.22	2.31	2.50
Diagonalize form	0.047	0.71	1.64	4.91	16.21	36.86	5.12
Detect orthogonality	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.20×10^{-3}	1.00×10^{-3}	0.062
Total	0.41	1.95	5.17	13.32	37.91	96.22	4.26

Table A2. Statistics of the matrix C and N with various values of n (execution time in seconds).

Step \ n	2	3	4	5	6	7	**Convergence**
C solving time	1.00×10^{-4}	1.30×10^{-3}	0.010	0.039	0.13	0.46	6.66
C dimension	13	71	261	736	1667	3613	4.50
C sparsity	1.00	0.67	0.47	0.40	0.27	0.28	-1.08
N solving time	0.055	0.068	0.10	0.15	0.33	0.58	1.85
N dimension	4	11	22	37	56	79	2.38

Table A3. Execution time (in seconds) of diagonalization with various values of d.

Step \ d	5	10	20	30	40	50	**Convergence**
Detect nondegeneracy	2.10×10^{-3}	7.30×10^{-3}	0.042	0.14	0.29	0.56	2.50
Compute the center	5.70×10^{-3}	0.027	0.18	0.66	1.76	3.98	2.85
Detect diagonalizability	0.034	0.033	0.032	0.032	0.031	0.033	-0.020
Compute idempotent	0.18	0.18	0.18	0.19	0.18	0.18	4.10×10^{-3}
Diagonalize form	0.45	0.97	4.28	29.62	109.25	398.64	2.93
Detect orthogonality	1.00×10^{-3}	1.00×10^{-3}	1.10×10^{-3}	1.20×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	-0.40
Total	1.58	3.85	13.69	50.16	145.09	454.08	2.39

where $d \in \mathbb{Z}^+$. We tested the computational costs of the diagonalization algorithm for various degrees d. We display the statistics for $d = 5, 10, 20, 30, 40,$ and 50 in Table A3, which includes the execution time and the convergence rates of each step and the entire program. The execution time of the program increases in the order of $O(d^{2.39})$, which is markedly lower compared with that for n in Example A.1. Among the six steps, the costs of the steps of computing the center and diagonalizing the form increase with near cubic order and are significantly higher than the other steps. The theoretical analysis shows that the complexity of the algorithm depends on t, which increases as d increases, as demonstrated in the experiment.

In contrast, the costs of the other four steps are similar for different values of d. The upper bounds of their complexity are $O(n^5)$, $O(\rho)$, and $O(n^3)$, respectively. Since d does not affect the dimension and solution time for the matrices B and N, it has a trivial influence on their execution time. This example also shows that the center-based diagonalization algorithm works for arbitrary d.

The numerical results in Examples A.1 and A.2 demonstrate that the diagonalization algorithm runs in polynomial time if either n or d is fixed. The convergence of each step of the algorithm is generally consistent with the complexity analysis based on the factors n, d, and t. In addition, the time-consuming steps like diagonalizing the form are highly parallelizable, and the parallel complexity of their polynomial multiplication can be reduced to $O(\log n)$ [16].

B. Number of polynomial terms

The number of terms t in the polynomial f is a critical factor in the complexity analysis of the diagonalization algorithm. We show that the criterion of diagonalizabiilty has polynomial-time complexity based on n, d, and t, with additional numerical results in Appendix A. An important question to be answered is the impact of t, which is affected by both n and d. The maximum number of terms t in f is bounded by $\binom{n+d-1}{d}$. When both n and t increase, the value of t grows exponentially. However, if either n or d is fixed, the growth becomes polynomial.

We investigated the growth of t values using a homogeneous polynomial f, where

$$f(x_1, x_2, \dots, x_n) = (x_1 + x_2 + \cdots + x_n)^d .$$

and $d \in \mathbb{Z}^+$. The values of t for various n and d values are shown in Tables A4 and A5, respectively. Table A5 shows that t increases when d is fixed and n increases. While the convergence rates increase for larger values of d, the growth is still polynomial. Similarly, t increases when n is fixed and d increases in Table A4.

Table A4. Number of terms t versus dimension n with a fixed degree d.

$d\backslash n$	2	3	4	5	6	7	8	9	10	**Convergence**
3	4	10	20	35	56	84	120	165	220	2.50
4	5	15	35	70	126	210	330	495	715	3.10
5	6	21	56	126	252	462	792	1287	2002	3.64
6	7	28	84	210	462	924	1716	3003	5005	4.12

Table A5. Number of terms t versus degree d with a fixed dimension n.

$n\backslash d$	2	3	4	5	6	7	8	9	10	**Convergence**
2	3	4	5	6	7	8	9	10	11	0.81
3	6	10	15	21	28	36	45	55	66	1.50
4	10	20	35	56	84	120	165	220	286	2.10
5	15	35	70	126	210	330	495	715	1001	2.64

Actual t values are often smaller than the theoretical upper bounds, as the coefficients of many terms may be zero. Operations like computing the Hessian matrix H for f may eliminate more terms, further reducing t, especially for a small d. Moreover, routines that depend on t are highly parallelizable. Therefore, the growth of t is polynomial when either n or d is fixed, and it will not jeopardize the performance of the diagonalization algorithm.