Decomposition of Ordinary Difference Polynomials

Mingbo Zhang

KLMM and Department of Mathematics
University of Science and Technology of China, Hefei Province, China

Xiao-Shan Gao

Key Laboratory of Mathematics Mechanization
Institute of Systems Science, AMSS, Chinese Academy of Sciences

Abstract

In this paper, we present an algorithm to decompose ordinary nonlinear difference polynomials with rational functions as coefficients. The algorithm provides an effective reduction of the decomposition of difference polynomials to the decomposition of linear difference polynomials over the same coefficient field. The algorithm is implemented in Maple for the constant coefficient case. Experimental results show that the algorithm is quite effective and can be used to decompose difference polynomials with thousands of terms.

Key words: Functional decomposition, difference polynomial, difference operator, difference degree.

1. Introduction

Functional decomposition for algebraic polynomials has been studied in detail and there exist complete theoretical (15) and efficient algorithmic (4; 5) results. In the 1950s, Ritt and his students established the differential algebra (14) and difference algebra (6) to deal with the differential and difference equations from the algebraic and algorithmic viewpoint. Naturally, the decomposition of differential polynomials and difference polynomials becomes a problem worth studying.

⋆ The work is partially supported by a National Key Basic Research Project of China (2004CB318000).
Email addresses: mbzhang@ustc.edu.cn (Mingbo Zhang), xgao@mmrc.iss.ac.cn (Xiao-Shan Gao).
URL: http://www.mmrc.iss.ac.cn/˜xgao (Xiao-Shan Gao).

Most existing work on decomposing differential and difference polynomials focuses on the linear case. Classical algorithms for factoring linear differential operators, such as Beke's algorithm and its improvements, can also be used in the difference case (1). Actually, linear differential and difference polynomials can be treated uniformly in the setting of Ore algebra (1; 2; 11; 12). In (9; 19; 20), partial results were given to decompose differential polynomials. In (10), a complete decomposition algorithm for nonlinear ordinary differential polynomials was given.

The main motivation for studying decomposition algorithms is to simplify the solution of differential or difference equations (17; 21). Theoretically, decomposition problems are closely related to various forms of Lüroth's theorem (16).

In this paper, we will give an algorithm to decompose nonlinear ordinary difference polynomials. Our algorithm consists of three steps. First, the problem of decomposing a general difference polynomial is reduced to the problem of decomposing a homogeneous one, which will be presented in Section 5. Second, the decomposition of a homogeneous difference polynomial is reduced to the construction of the linear left decomposition factors of another homogenous difference polynomial, which will be presented in Section 4. Finally, construction of linear left decomposition factors of a homogenous difference polynomial is reduced to the decomposition of linear difference polynomials, which will be presented in Section 3. As mentioned above, there exist decomposition algorithms for linear difference polynomials (1; 2; 11).

The worst case complexity of the algorithm is exponential. The reason is due to the combinatorial selections in the algorithm. Notice that the complexity of decomposing linear difference polynomials is also exponential. Despite of the exponential complexity, the algorithm is quite effective practically. There are two reasons for this. First, all the computations in the algorithm use explicit formulas to compute the result, and as a consequence, for each of the combinatorial selection, the computation is extremely fast. Second, in most practical examples, the number of possible combinatorial selections is not very large. The experimental results also support this assertion. Similar to the differential case (10), our algorithm presented in this paper does not introduce new parameters during the decomposition process and works in the coefficient field of the given difference polynomial.

The algorithm is implemented in Maple in the constant coefficient case [1]. Extensive experiments show that the program can be used to decompose difference polynomials with thousands of terms effectively. The results can found in Section 7.

Comparing to decomposition algorithms for algebraic polynomials, our algorithm is a generalization of the algorithm by Kozen and Landau (4) to ordinary difference polynomials. We need to consider much more cases than the algorithm in (4) due to the introduction of the difference operators. The decomposing algorithm for difference polynomials presented in this paper is quite different from its differential counterpart presented in (10). This is mainly due to the distinct properties of the differential and difference operators. Let y and z be indeterminants, δ a difference operator, and ∂ a differential operator. Then, we have $\delta(y^2 + yz) = \delta(y)^2 + \delta(y)\delta(z)$ and $\partial(y^2 + yz) = 2y\partial(y) + \partial(y)z + y\partial(z)$. From this, we can see that the "structure" of a difference polynomial does not change after the action of a difference operator, which is not the case for differential polynomials.

[1] The package is available at http://www.mmrc.iss.ac.cn/~ xgao/software/dec-chafen.zip

Due to this property, our algorithm for decomposing difference polynomials is simpler and more efficient than its differential counter part.

The rest of the paper is organized as follows. In Section 2, notations and preliminary results are presented. In Sections 3, 4, and 5, we present the three major steps of the algorithm. In Section 6, we show how to find decomposition factors containing parameters. Experimental results are given in Section 7. We conclude the paper in Section 8.

2. Notations and Preliminary Results

Let $\mathcal{K} = \mathbb{Q}(x)$ be the field of rational functions in x, δ a difference operator over $\mathcal{K}$ such that $\delta(r) = r$ for $r \in \mathbb{Q}$, $\delta(x) \neq x$, and for any $s \in \mathcal{K}$ there exists a $t \in \mathcal{K}$ such that $\delta(t) = s$. In other words, $\mathcal{K}$ is a reflexive difference field (6). For instance, we may take $\delta(x) = x + 1$ or $\delta(x) = q \cdot x$ for $q \in \mathbb{Q} \setminus \{0\}$ as the difference operator.

For a difference indeterminate y, let $\mathcal{K}\{y\}$ be the ordinary difference polynomial ring over $\mathcal{K}$ (6). An element in $\mathcal{K}\{y\}$ is called a *difference polynomial.* We denote by $y_i = \delta^i y$ the i-th transform of y. Let $f \in \mathcal{K}\{y\} \setminus \mathcal{K}$. The largest i such that y_i appears in f is called the *order* of f, denoted by o_f. We can write f as the following form

$$f = f_d y_{o_f}^d + f_{d-1} y_{o_f}^{d-1} + \cdots + f_0 \tag{1}$$

where f_i is an algebraic polynomial in $y, y_1, y_2, \ldots, y_{o_f - 1}$ and $f_d \neq 0$. We call $d_f \triangleq d$ the *degree* of f and $i_f \triangleq f_d$ the *initial* of f. We can also write f as

$$f = \sum a_{i_0 i_1 \cdots i_{o_f}} y^{i_0} y_1^{i_1} \cdots y_{o_f}^{i_{o_f}}$$

where each $a_{i_0 i_1 \cdots i_{o_f}} \in \mathcal{K}$. We call $a_{i_0 i_1 \cdots i_{o_f}} y^{i_0} y_1^{i_1} \cdots y_{o_f}^{i_{o_f}}$ a *term* of f.

$$\max\{i_0 + i_1 + \cdots + i_{o_f} \,|\, a_{i_0 i_1 \cdots i_{o_f}} \neq 0\}$$

is called the *total degree* of f, which is denoted by $\mathrm{tdeg}(f)$.

$$\max\{i_1 + 2i_2 + \cdots + o_f i_{o_f} \,|\, a_{i_0 i_1 \cdots i_{o_f}} \neq 0\}$$

is called the *difference degree* of f and is denoted by $\mathrm{ddeg}(f)$.

If the total degrees of all the terms in f are the same, f is called *homogeneous*; furthermore, if the difference degrees of all the terms in f are equal, f is called *difference homogeneous.* In particular, if f is homogeneous and its total degree equals one, f is called *linear.*

We may define a rank between two terms according to the pure lexicographical order induced by the variable order $y < y_1 < y_2 < \cdots$. In a difference polynomial f, the term with the highest rank is called the *leading term* of f and is denoted by l_f. A difference polynomial f is said to be of *lower rank* than that of g if the leading term of f is of lower rank than that of g.

Let $g, h \in \mathcal{K}\{y\}$. We use $g \circ h$ to denote the *functional composition* of g and h, which is defined by substituting y_i in g with the i-th transform of h. If $f = g \circ h$, g and h are called the *left* and *right decomposition factors* of f respectively. A *decomposition* $f = g \circ h$ is called *nontrivial*, if both g and h are not of the form $ay + b$, where a and b are in $\mathcal{K}$. Two decompositions $f = g_1 \circ h_1$ and $f = g_2 \circ h_2$ are called *equivalent* if there exist $a, b \in \mathcal{K}$ such that $h_1 = (ay + b) \circ h_2$. A decomposition $f = p_1 \circ p_2 \cdots \circ p_n$ is called a *maximal decomposition* of f, if there is no nontrivial decomposition for each p_i. In this paper, we only consider how to compute a decomposition of length two and we are only interested in the nontrivial and nonequivalent decompositions. Let us see two examples of decompositions of difference polynomials.

Example 1. $(x+1)y_1y_2^3y_3^2 + xyy_1^3y_2^2 = \big((x+1)y_1y_2 + xyy_1\big) \circ yy_1^2$ is a decomposition over the field $\mathbb{Q}(x)$.

Notice that the decomposition factors of linear difference polynomials may contain parameters as shown by the following example.

Example 2. Let the difference operator be: $\delta(x) = x+1$.

$$y_2 - 3y_1 + 2y = \left(y_1 - \frac{c\cdot 2^x - 2}{c\cdot 2^x - 1}y\right) \circ \left(y_1 - \frac{2(c\cdot 2^x - 1)}{c\cdot 2^x - 2}y\right)$$

is a decomposition of $y_2 - 3y_1 + 2y$ in $\mathbb{Q}(x, 2^x)$, where c is any difference constant.

In Sections 3, 4 and 5, we will give algorithms to find decomposition factors for a difference polynomial, which do not contain parameters like the one in Example 2. In Section 6, we will show how to modify our algorithms to find decomposition factors with parameters.

The following properties of the difference decomposition can be easily verified.

Lemma 3. *The composition operation is associative:* $f\circ(g\circ h) = (f\circ g)\circ h$.

Lemma 4. *If* $f = g\circ h$ *is a decomposition of* f*, then* $o_f = o_g + o_h$, $d_f = d_g \cdot d_h$*, and* $i_f = (i_g \circ h)\cdot (i_h)_{o_g}^{d_g}$.

By Lemma 3, for any $c \in \mathcal{K}$, we have $f = g\circ h = \bigg(g\circ(y+c)\bigg)\circ\bigg((y-c)\circ h\bigg)$. So we can assume that h has no term in $\mathcal{K}$. In this case, the term of f in $\mathcal{K}$ is equal to that of g. So we will assume that *f, g, and h have no terms in $\mathcal{K}$ in the rest of this paper.*

In this paper, we will show how to find a nontrivial decomposition $f = g\circ h$ for a given difference polynomial f. After obtaining the right decomposition factor of f, we can compute the corresponding left decomposition factor easily. One possible way is to estimate the total degree and the order of g with Lemma 4 and then find the coefficients of g by solving a algebraic linear equation system. The following algorithm gives a more direct solution to this problem.

Algorithm 1. Input: difference polynomials f and h.

Output: a difference polynomial g such that $f = g\circ h$ if such a g exists.

S1 If $f = 0$, then return $g = 0$. Let $o_g = o_f - o_h$, $d_g = d_f/d_h$. By Lemma 4, if $o_g < 0$ or d_g is not an integer, g does not exist and the algorithm terminates.

S2 Suppose that $g = i_g y_{o_g}^{d_g} + u_g$. We will find i_g and u_g separately.

S3 By Lemma 4, $i_g \circ h = i_f/(i_h)_{o_g}^{d_g}$. If $q = i_f/(i_h)_{o_g}^{d_g}$ is not a difference polynomial, g does not exist and the algorithm terminates. Otherwise, execute Algorithm 1 with q

and h as input and i_g as output. If i_g does not exist, g does not exist and the algorithm terminates.

S4 Let $f_1 = f - (i_g y_{o_g}^{d_g}) \circ h$. Then $f_1 = g \circ h - (i_g y_{o_g}^{d_g}) \circ h = (g - i_g y_{o_g}^{d_g}) \circ h = u_g \circ h$. Call Algorithm 1 with f_1 and h as input and u_g as output. If u_g exists, output $i_g y_{o_g}^{d_g} + u_g$; otherwise g does not exist.

The algorithm clearly terminates since q and f_1 are of lower ranks than that of f. Due to Dickson's Lemma (7), a sequence of difference polynomials with strictly decreasing ranks must be finite.

In the rest of this paper, we will concentrate on computing the right decomposition factor of a given difference polynomial.

3. Linear Left Decomposition Factor

In this section, we will solve the following problem:

Problem L. *For a homogeneous difference polynomial f, compute a decomposition $f = g \circ h$ such that g is linear.*

The key idea is that we can construct a linear difference polynomial L such that if g is a linear left decomposition factor of f, then g must be a left factor of L. This reduces the problem to decomposition of linear difference polynomials.

In the rest of this section, we will assume that f and h are homogeneous and g is linear. Notice that when f, g, h satisfy the conditions of Problem L, f and h have the same total degree. We first prove the following lemma.

Lemma 5. *If* $\mathrm{ddeg}(f) < \mathrm{tdeg}(f)$*, then f has no nontrivial decomposition $f = g \circ h$ such that g is linear.*

Proof. Let $u = a_I y_l^{i_l} \cdots y_m^{i_m}$ be any term of f, where l is the minimal integer such that y_l appears in u. If $l \geq 1$, then $\mathrm{ddeg}(f) \geq \mathrm{ddeg}(u) = l \cdot i_l + \cdots + m \cdot i_m \geq i_l + \cdots + i_m = \mathrm{tdeg}(u) = \mathrm{tdeg}(f)$ (recall that f and h are assumed to be homogeneous in this section), which contradicts to $\mathrm{ddeg}(f) < \mathrm{tdeg}(f)$. So y must appear in each term of f. But for a non-trivial decomposition $f = g \circ h$ where g is linear, we have $o_g > 0$ and $g \circ h$ must contain a term which does not involve y, a contradiction. ∎

Let $a = \mathrm{ddeg}(f), \alpha = \mathrm{tdeg}(f)$ and write f as the sum of difference homogeneous parts:

$$f = F_a + F_{a-1} + \cdots + F_0$$

where F_j is the sum of terms in f with difference degree j $(0 \leq j \leq a)$. Let $k = a \bmod \alpha$ and $a = t \cdot \alpha + k$. By Lemma 5, we have $t \geq 1$. Let

$$L_i(y, z) = F_{a-i} z_t + F_{a-\alpha-i} z_{t-1} + \cdots + F_{a-t\alpha-i} z \ , i = 0, \ldots, k \tag{2}$$

Then L_i is a linear difference polynomial in z with coefficients in $\mathcal{K}\{y\}$.

Lemma 6. *If a linear difference polynomial $g \in \mathcal{K}\{y\}$ is a left decomposition factor of f, then it must be a left decomposition factor of each L_i for $i = 0, \ldots, k$.*

Proof. Let $g = c_n y_n + \cdots + c_0 y \in \mathcal{K}\{y\}$ be a linear left decomposition factor of f and $h = H_b + H_{b-1} + \cdots + H_0$ be the corresponding right decomposition factor, where $b = \mathrm{ddeg}(h)$ and H_l is the sum of terms with difference degree $l (0 \le l \le b)$. Then

$$F_a + F_{a-1} + \cdots + F_0 = (c_n y_n + c_{n-1} y_{n-1} + \cdots + c_0 y) \circ (H_b + H_{b-1} + \cdots + H_0). \quad (3)$$

It is clear that $F_a = c_n y_n \circ H_b$, from which we can derive that $a = b + n\alpha$. For a fixed $0 \le i \le k$, by comparing the parts with difference degree $a - j\alpha - i = b - i + (n-j)\alpha$ $(0 \le j \le t)$ on both sides of (3), we have

$$F_{a-j\alpha-i} = c_n y_n \circ H_{b-i-j\alpha} + c_{n-1} y_{n-1} \circ H_{b-i-(j-1)\alpha} + \cdots + c_{n-j} y_{n-j} \circ H_{b-i}.$$

While the coefficient of z_{t-j} in

$$(c_n z_n + \cdots + c_0 z) \circ (H_{b-i} z_{t-n} + H_{b-i-\alpha} z_{t-n-1} + \cdots + H_{b-i-(t-n)\alpha} z)$$

is $c_n y_n \circ H_{b-i-j\alpha} + c_{n-1} y_{n-1} \circ H_{b-i-(j-1)\alpha} + \cdots + c_{n-j} y_{n-j} \circ H_{b-i} = F_{a-j\alpha-i}$. From (2), we have

$$L_i = (c_n z_n + \cdots + c_0 z) \circ (H_{b-i} z_{t-n} + H_{b-i-\alpha} z_{t-n-1} + \cdots + H_{b-i-(t-n)\alpha} z) \quad (4)$$

Namely, g is a left decomposition factor of L_i. ∎

By Lemma 6, we can first compute the common left decomposition factors of the linear difference polynomials $L_i, i = 0, \ldots, k$ and check whether they are left decomposition factors of f. Notice that L_i also involves the difference variable y, which can be eliminated by using the following result.

Lemma 7. *If $g \in \mathcal{K}\{y\}$ is a linear left decomposition factor of f, then we can choose $\gamma_i \in \mathcal{K}$ such that $L_i(\gamma_i, z) \neq 0$ and g is a left decomposition factor of*

$$\bar{L_i} = L_i(\gamma_i, z) = F_{a-i}(\gamma_i) z_t + F_{a-\alpha-i}(\gamma_i) z_{t-1} + \cdots + F_{a-t\alpha-i}(\gamma_i) z, i = 0, \ldots, k.$$

Proof. By Lemma II on page 201 of (6), we can always choose a $\gamma_i \in \mathcal{K}$ such that $L_i(\gamma_i, z) \neq 0$ for each i. Since $g = c_n y_n + \cdots + c_0 y$ is a left decomposition factor of f, by Lemma 6, we have equation (4). Replacing y with $\gamma_i \in \mathcal{K}$ in (4), we have

$$L_i(\gamma_i, z) = (c_n z_n + \cdots + c_0 z) \circ (H_{b-i}(\gamma_i) z_{t-n} + \cdots + H_{b-(t-n)\alpha-i}(\gamma_i) z)$$

So, g must be a linear left decomposition factor of $\bar{L_i}$. ∎

By Lemma 7, to solve Problem L, we need only to find all the *common left decomposition factors* of the linear difference polynomials $\bar{L_i} \in \mathcal{K}\{z\}, i = 0, \ldots, k$ and check whether they are left decomposition factors of f.

It is clear that decomposing a linear difference polynomial is equivalent to factoring a linear difference operator. Most of the algorithms for factoring differential operators can be used for the difference case, such as Beke's algorithm (1).

Based on the above analysis, we give the following algorithm to solve Problem L.

Algorithm 2. Input: a homogeneous difference polynomial f and a positive integer n.

Output: a set S of all possible (g, h) such that $f = g \circ h$ is a nontrivial decomposition of f, $o_g = n$ and g is linear.

S1 Let $S := \{\}$. If y appears in every term of f, by Lemma 6, f has no linear decomposition factors. Return S.

S2 Let $\alpha = \text{tdeg}(f), a = \text{ddeg}(f), t = \lfloor \frac{a}{\alpha} \rfloor$, and $a = t \cdot \alpha + k$. If $a < n \cdot \alpha$, then g and h do not exist and the algorithm terminates. Otherwise, write f as the sum of the difference degree homogeneous parts: $f = F_a + F_{a-1} + \cdots + F_0$ and let $L_i(y, z) = F_{a-i}z_t + F_{a-\alpha-i}z_{t-1} + \cdots + F_{a-t\alpha-i}z, i = 0, \ldots, k$.

S3 For each L_i, choose a $\gamma_i \in \mathcal{K}$ such that $\bar{L}_i = L_i(\gamma_i, z) \neq 0, i = 0, \ldots, k$.

S4 Compute the greatest common left divisor $\tilde{L}$ of $\bar{L}_i, i = 0, \cdots, k$ with the left Euclidean remainder sequence method (1; 12).

S5 Compute the left decomposition factors of $\tilde{L}$ of order n with algorithms in (1; 2; 11; 12).

S6 For each g obtained in S5, check whether g is a left decomposition factor of f with Algorithm 3. If it is, compute the correspondent right decomposition factor h and add (g, h) to S, return S. Note that g and h may contain parameters, this issue will be addressed later in Section 6.

Note that in Step **S3**, the coefficients of the linear difference polynomial $\bar{L}_i$ are still in $\mathcal{K}$. In other words, we do not need to do the decomposition in an extension field.

Algorithm 3. Input: a difference polynomial f and a linear difference polynomial g.
Output: a difference polynomial h such that $f = g \circ h$ if such an h exists.

S1 Let $h := 0$.

S2 Since g is linear, by Lemma 4 we have $o_h = o_f - o_g$, $d_h = d_f$, $i_h = y_{-o_g} \circ (i_f/i_g)$. If i_h is not a difference polynomial, then h does not exist and the algorithm terminates.

S3 Let $h := h - i_h y_{o_h}^{d_h}$, $f := f - g \circ (i_h y_{o_h}^{d_h})$. If $f = 0$, output h; otherwise, go to S2.

In step S2, we use $y_{-o_g} \circ p$ to represent the following *inversion procedure*. Let m be the minimal integer such that y_m appears in p and $p = \sum a_{i_m \cdots i_{o_f}} y_m^{i_m} \cdots y_{o_f}^{i_{o_f}}$. If $o_g > m$, $y_{-o_g} \circ p$ does not exist; otherwise $y_{-o_g} \circ p = \sum (\delta^{-o_g} a_{i_m \cdots i_{o_f}}) y_{m-o_g}^{i_m} \cdots y_{o_f - o_g}^{i_{o_f}}$.

Algorithm 3 is correct because g is linear and hence $g \circ (i_h y_{o_h}^{d_h} + u_h) = g \circ (i_h y_{o_h}^{d_h}) + g \circ u_h$.

Remark 8. In Steps S3 and S4 of Algorithm 2, we may use the following strategy to improve the computational efficiency. We may select more values $\eta_i \in \mathcal{K}, i = 0, \ldots, k$ such that $L_i' = L_i(\eta_i, z) \neq 0, i = 0, \ldots, k$. Let $\tilde{L}'$ be the greatest common left divisor of $\bar{L}_i, L_i', i = 0, \cdots, k$. Then $\tilde{L}'$ could be of lower order than that of $\tilde{L}$. In Step S5, we can compute the factors of $\tilde{L}'$ in stead of $\tilde{L}$.

Example 9. Let the difference operator be $\delta(x) = x + 1$. Let $f = y_2 y_3 + y_3^2 + y_2 y_4 + y y_1 + y_1^2 + y y_2$ be a homogeneous difference polynomial over $\mathcal{K}$. We use Algorithm 2 to compute its linear left decomposition factors of order two.

In S2, we have $\alpha = \text{tdeg}(f) = 2$, $a = \text{ddeg}(f) = 6$, $k = \mod(a, \alpha) = 0$, and $f = F_6 + F_5 + F_2 + F_1 = (y_3^2 + y_2 y_4) + (y_2 y_3) + (y y_2 + y_1^2) + (y y_1)$. By Lemma 6, if $g(y)$ is a linear left decomposition factor of f, $g(z)$ must be a linear left decomposition factor of $L_0 = (y_3^2 + y_2 y_4) z_3 + (y y_2 + y_1^2) z_1$.

In S3, if we choose $y = 1$, we have that $g(z)$ is a linear left decomposition factor of $L_0(1, z) = 2z_3 + 2z_1$. It is clear that $g(z) = z_2 + z$ and $h = 2z_1$. If we choose $y = x$, then $L_0(x, z) = (2x^2 + 12x + 17)z_3 + (2x^2 + 4x + 1)z_1$. The only non-equivalent linear left decomposition factor of $L_0(x, z)$ with order two is $g(z) = z_2 + z$ $(L_0(x, z) = (z_2 +$

$z) \circ \big((2x^2+4x+1)z_1\big)\big)$. In S6, we can check that $g(y) = y_2 + y$ is really a linear left decomposition factor of f: $f = (y_2 + y) \circ (yy_2 + y_1^2 + yy_1)$.

4. Decomposition of Homogeneous Difference Polynomials

In this section, we consider the following decomposition problem.
Problem H. Let f be a homogeneous difference polynomial. Find all the decomposition factors of f.

The idea is first reducing the problem to the decomposition of another difference polynomial f', the initial of whose left decomposition factor must be in $\mathcal{K}$ and then solving this new problem directly. We first prove two lemmas.

Lemma 10. *If $f = g \circ h$, then f_{d_f-i} is divisible by $y_{o_g} \circ h_{d_h}^{d_g-i} (0 \le i \le d_g)$, where f_j is the coefficient of $y_{o_f}^j$ in f.*

Proof. If $f = g \circ h$, we write f, g, h as their canonical representations similar to (1)

$$\begin{aligned} f &= f_{d_f} y_{o_f}^{d_f} + f_{d_f-1} y_{o_f}^{d_f-1} + \ldots + f_1 y_{o_f} + f_0 \\ g &= g_{d_g} y_{o_g}^{d_g} + g_{d_g-1} y_{o_g}^{d_g-1} + \ldots + g_1 y_{o_g} + g_0 \\ h &= h_{d_h} y_{o_h}^{d_h} + h_{d_h-1} y_{o_h}^{d_h-1} + \ldots + h_1 y_{o_h} + h_0. \end{aligned} \tag{5}$$

We have

$$\begin{aligned} g \circ h = {} & (g_{d_g} \circ h) \cdot \big[(y_{o_g} \circ h_{d_h}) y_{o_g+o_h}^{d_h} + \cdots + y_{o_g} \circ h_0\big]^{d_g} + (g_{d_g-1} \circ h) \cdot \\ & \big[(y_{o_g} \circ h_{d_h}) y_{o_g+o_h}^{d_h} + \cdots + y_{o_g} \circ h_0\big]^{d_g-1} + \cdots + g_0 \circ h \end{aligned}$$

Comparing the coefficients of $y_{o_f}^i (d_f \ge i \ge d_f - d_h - 1)$ on both sides of $f = g \circ h$, we have

$$\begin{aligned} f_{d_f} &= (g_{d_g} \circ h) \cdot y_{o_g} \circ h_{d_h}^{d_g} \\ f_{d_f-1} &= (g_{d_g} \circ h) \cdot y_{o_g} \circ [d_g h_{d_h}^{d_g-1} h_{d_h-1}] \\ & \cdots \\ f_{d_f-i} &= (g_{d_g} \circ h) \cdot y_{o_g} \circ [d_g h_{d_h}^{d_g-1} h_{d_h-i} + T_i] \\ & \cdots \\ f_{d_f-d_h} &= (g_{d_g} \circ h) \cdot y_{o_g} \circ [d_g h_{d_h}^{d_g-1} h_0 + T_{d_h}] + (g_{d_g-1} \circ h) \cdot (y_{o_g} \circ h_{d_h}^{d_g-1}) \end{aligned} \tag{6}$$

where

$$T_i = \boxed{(h_{d_h} y_{o_h}^{d_h} + h_{d_h-1} y_{o_h}^{d_h-1} + \cdots + h_{d_h-i+1} y_{o_h}^{d_h-i+1})^{d_g}}_{\,d_f-i} \tag{7}$$

denotes the coefficient of $y_{o_h}^{d_f-i}$ in $(h_{d_h} y_{o_h}^{d_h} + h_{d_h-1} y_{o_h}^{d_h-1} + \cdots + h_{d_h-i+1} y_{o_h}^{d_h-i+1})^{d_g}$ $(1 \le i \le d_h)$. Notice that T_i is divisible by $h_{d_h}^{d_g-i}$, so we prove the lemma. ∎

Lemma 11. *If $f = g \circ h$, then h is also a right decomposition factor of the difference polynomials $f^{(k)}, k = 1, \ldots, s$ which are defined as: $f^{(0)} = f$, $g^{(0)} = g$, and*

$$g^{(k)} = i_{g^{(k-1)}}, \quad f^{(k)} = \frac{i_{f^{(k-1)}}}{y_{o_{g^{(k-1)}}} \circ i_h^{d_{g^{(k-1)}}}} = g^{(k)} \circ h, k = 1, \ldots, s. \tag{8}$$

Proof. By the first equation in (6), we have

$$f^{(1)} = \frac{i_f}{y_{o_g} \circ i_h^{d_g}} = \frac{f_{d_f}}{y_{o_g} \circ h_{d_h}^{d_g}} = g_{d_g} \circ h = g^{(1)} \circ h$$

which proves the case: $k = 1$. By repeating this process, we may prove other cases. Note that these results are true under the condition that $f^{(k)}$ are difference polynomials. ▮

Before solving Problem H, we first solve the following simpler problem.
Problem H1. Let F be a homogeneous difference polynomial. Find the decompositions $f = g \circ h$ such that (d_h, o_h, h_{d_h}) equals to the given value and the initial of g is in $\mathcal{K}$.

We consider two cases:

(1) $d_g = 1$. Since g is homogeneous, g is linear. We can compute g, h with Algorithm 2.

(2) $d_g > 1$. From the first equation in (6), we have

$$g_{d_g} = \frac{f_{d_f}}{y_{o_g} \circ h_{d_h}^{d_g}}. \tag{9}$$

We can do this, because $g_{d_g} \in \mathcal{K}$ and hence $g_{d_g} \circ h = g_{d_g}$. We can obtain h_i for $i = 1, \ldots, d_h - 1$ from the second to the $(d_h - 1)$-th equations in (6) as follows

$$h_{d_h - i} = (y_{-o_g} \circ \frac{f_{d_f - i}}{g_{d_g}} - T_i)/(d_g h_{d_h}^{d_g - 1}), i = 1, \ldots, d_h - 1. \tag{10}$$

To compute h_0, from the last formula in (6), we have

$$d_g g_{d_g} \cdot y_{o_g} \circ h_0 + g_{d_g - 1} \circ h = \left(f_{d_f - d_h} - g_{d_g} \cdot y_{o_g} \circ T_{d_h} \right) \Big/ y_{o_g} \circ h_{d_h}^{d_g - 1} \triangleq F_1 \tag{11}$$

Substituting $h_0 = h - \sum\limits_{1 \le i \le d_h} h_i y_{o_h}^i$ into (11), we have

$$(d_g g_{d_g} y_{o_g} + g_{d_g - 1}) \circ h = F_1 + d_g g_{d_g} y_{o_g} \circ (\sum_{1 \le i \le d_h} h_i y_{o_h}^i) \triangleq F_2 \tag{12}$$

Note that $g_{d_g - 1}$ is a linear difference polynomial with order less than o_g and F_2 can be computed. Then $d_g g_{d_g} y_{o_g} + g_{d_g - 1}$ is a linear left decomposition factor of F_2 with order o_g and we can obtain all possible h by executing Algorithm 2 with input $(F_2,\ o_g)$.

Now, we can describe our algorithm in three steps.

- By Lemmas 4 and 10, the three elements (d_h, o_h, h_{d_h}) satisfy $d_h \mid d_f$, $o_h \leq o_f$, $(y_{o_g} \circ h_{d_h}^{d_g-i})$ divides $f_{d_f-i}(0 \leq i \leq d_g)$. From these equations, we may guess all possible values for (d_h, o_h, h_{d_h}) from f.
- After (d_h, o_h, h_{d_h}) is known, by Lemma 4, we have $d_g = \frac{d_f}{d_h}$, $o_g = o_f - o_h$. Repeat (8) until either $f^{(k)}$ is not a difference polynomial or $f^{(k+1)} \in \mathcal{K}$. In the first case, there exists no h satisfying the condition. In the second case, we solve Problem H1 as mentioned above.
- Finally, we check whether the right decomposition factors found in the preceding step are the right decomposition factor of f.

Based on the analysis above, we propose the following algorithm to compute all the decompositions of a given homogeneous difference polynomial.

Algorithm 4. Input: a homogeneous difference polynomial $f \in \mathcal{K}\{y\}$.

Output: $T = \{(g, h)\}$ is the set of all non-equivalent decomposition factors of f.

S1 $T := \{\}$, $\bar{f} := f$. Let $S = \{(d_h, o_h, h_{d_h}) : d_h | d_f,\ 0 \leq o_h \leq o_f,\ y_{o_f-o_h} \circ h_{d_h}^{d_f/d_h-i}$ divides $f_{d_f-i}(0 \leq i \leq d_g - 1)\}$, where f_i is the coefficient of $y_{o_f}^i$ in f.

S2 If S is empty, then return T; otherwise, choose a (d_h, o_h, h_{d_h}) in S and let $S := S \setminus \{(d_h, o_h, h_{d_h})\}$. Let $\bar{f} = f$.

S3 Let $d_g := \frac{d_{\bar{f}}}{d_h}$, $o_g := o_{\bar{f}} - o_h$, and $\bar{f}_i$ the coefficient of $y_{o_{\bar{f}}}^i$ in $\bar{f}$. If d_g is not an integer, or $o_g < 0$, or there exists an i $(0 \leq i \leq d_g)$ such that $y_{o_g} \circ h_{d_h}^{d_g-i}$ does not divide $\bar{f}_{d_{\bar{f}}-i}$, then there exists no h corresponding to (d_h, o_h, h_{d_h}) and go to S2; otherwise, go to S4.

S4 Let $\hat{f} := \frac{\bar{f}_{d_{\bar{f}}}}{y_{o_g} \circ h_{d_h}^{d_g}}$. If $\hat{f} \notin \mathcal{K}$, then $\bar{f} := \hat{f}$ and go to S3; otherwise, go to S5. This step corresponds to (8).

S5 If $d_g = 1$, then g is linear and execute Algorithm 2 with input $\bar{f}, o_g$. Let the output be the set G, go to S7.

S6 If $d_g > 1$, compute g_{d_g} with (9) and compute $h_{d_h-i}, i = 1, \ldots, d_h - 1$, F_1 and F_2 with (10), (11), and (12) respectively. In the above computation, if one of the results is not a difference polynomial, then go to S2. Execute Algorithm 2 with input F_2, o_g. Let the output be G.

S7 For each (g, h) in G, check whether h is a right decomposition factor of f with Algorithm 1. If it is and the corresponding left decomposition factor is g', then add (g', h) to T. Go to S2.

Example 12. Let $f = (x+1)y_1y_2^3y_3^2 + xyy_1^3y_2^2 \in \mathcal{K}\{y\}$. We compute all the decomposition factors of f.

In step S1, we have $S = \{(1, 1, 1), (1, 2, 1), (1, 3, 1), (1, 1, y), (1, 2, y_1), (1, 3, y_2), (2, 3, 1), (2, 3, y_2), (2, 3, y_2^2), (2, 3, y_2^3), (2, 3, y_1), (2, 3, y_1y_2), (2, 3, y_1y_2^2), (2, 3, y_1y_2^3), (2, 2, 1), (2, 2, y_1), (2, 2, y_1^2), (2, 2, y_1^3), (2, 2, y), (2, 2, yy_1), (2, 2, yy_1^2), (2, 2, yy_1^3), (2, 1, 1), (2, 1, y), (2, 1, y^2), (2, 1, y^3)\}$.

In step S2, we choose $(2, 1, y) \in S$ to start the computation. Steps S3 and S4 run three times continually and output $\bar{f} = (x + 1)y_2y_3^2$ $(d_g = 1)$. In step S5, execute Algorithm 2 to compute the second order linear left decomposition factor of $\bar{f} = (x + 1)y_2y_3^2$, the output is $\big(y_2, (x-1)yy_1^2\big)$. In step S7, we can check that $f = (\frac{y_1y_2}{x} + \frac{yy_1}{x-1}) \circ \big((x-1)yy_1^2\big)$.

If we choose other elements in S to start in step S2, only $(2, 2, yy_1^3)$ leads to the decomposition $f = (y_1 + y) \circ (xyy_1^3 y_2^2)$. Here we omit the details. Note that $(\frac{y_1 y_2}{x} + \frac{yy_1}{x-1}) \circ \big((x-1)yy_1^2\big) = \big((x+1)y_1y_2 + xyy_1\big) \circ yy_1^2$ are two equivalent decompositions.

From the above example, we can see that there could exist many choices for (d_h, o_h, h_{d_h}), but most of them do not lead to a decomposition. This may reduce the efficiency. How to improve the algorithm on this aspect is one of our future research topics.

5. Decomposition in the General Case

We now consider the decomposition algorithm in the general case, which is based on the following result.

Theorem 13. *Let $f = g \circ h$ be a nontrivial decomposition of f, t, a, b the total degrees of f, g, h respectively, and*

$$\begin{aligned} f &= F_t + F_{t-1} + \cdots + F_1 \\ g &= G_a + G_{a-1} + \cdots + G_1 \\ h &= H_b + H_{b-1} + \cdots + H_1 \end{aligned}$$

the representations of f, g, h as sums of homogeneous parts respectively. Then $F_t = G_a \circ H_b$ and $H_i (1 \le i \le b-1)$ can be determined explicitly and uniquely from H_b, G_a and f.

Now, we can give the general framework for our main algorithm. For a given difference polynomial f, we write f as the sum of its homogeneous parts $f = F_t + F_{t-1} + \cdots + F_1$. With Algorithm 4, we can find all p and q such that $F_t = p \circ q$. Then we will use q as a candidate for H_b to compute h based on Theorem 13. Finally, g can be computed with Algorithm 1.

Before proving Theorem 13, we need to give two lemmas and a crucial sub-algorithm.

Lemma 14. *Let u, v, h, p be difference polynomials. We denote the leading term of a difference polynomial f by l_f. Then we have*

(1) $l_{u \cdot v} = l_u \cdot l_v$.

(2) $l_{u \circ v} = l_u \circ l_v$.

(3) If y_j appears in l_u, then $l_{\frac{\partial u}{\partial y_j}} = \frac{\partial l_u}{\partial y_j}$.

(4) For a nonnegative integer j, if y_j appears in l_u, the leading term of $\frac{\partial u}{\partial y_j} \circ h \cdot y_j \circ p$ is

$$\frac{\partial l_u}{\partial y_j} \circ l_h \cdot y_j \circ l_p = \beta_j l_u \circ l_h \cdot y_j \circ \frac{l_p}{l_h}$$

where $\beta_j = \deg_{y_j} l_u$ is the degree of l_u in y_j.

Proof. The first three properties can be easily proved and the fourth one can be deduced from the first three. ∎

Lemma 15. *Let f, g, and h be homogeneous difference polynomials. If a difference polynomial p satisfies*

$$f = \sum_{0 \le j \le o_g} \frac{\partial g}{\partial y_j} \circ h \cdot y_j \circ p \tag{13}$$

and $\text{tdeg}(p) < \text{tdeg}(h)$*, then p is unique and can be computed from f, g and h explicitly.*

Proof. We use rank(f)>rank(g) to denote the fact that the rank of f is higher than the rank of g. Assume that there is a p satisfying the conditions in the lemma and q is an arbitrary term included in g. Let

$$l_g = \alpha y_k^{a_k} y_{k+1}^{a_{k+1}} \cdots y_{o_g}^{a_{o_g}},\ q = \beta y_n^{b_n} y_{n+1}^{b_{n+1}} \cdots y_m^{b_m}$$

where k is the least integer such that y_k appears in l_g and n, m are the minimal and maximal integers such that y_n, y_m appear in q. We divide the problem into two cases.
(1). rank(p)> rank(h). By Lemma 14, the leading term of $\sum_{0 \le j \le o_g} \frac{\partial q}{\partial y_j} \circ h \cdot y_j \circ p$ is

$$\frac{\partial q}{\partial y_m} \circ l_h \cdot y_m \circ l_p = b_m q \circ l_h \cdot y_m \circ \frac{l_p}{l_h}.$$

Since rank(l_g)≥rank(q), $o_g \ge m$, and rank(p)>rank(h), the rank of $a_{o_g} l_g \circ l_h \cdot y_{o_g} \circ \frac{l_p}{l_h}$ is higher than that of $b_m q \circ l_h \cdot y_m \circ \frac{l_p}{l_h}$. Then the leading term of $\sum_{0 \le j \le o_g} \frac{\partial g}{\partial y_j} \circ h \cdot y_j \circ p$ is

$$\frac{\partial l_g}{\partial y_{o_g}} \circ l_h \cdot y_{o_g} \circ l_p = a_{o_g} l_g \circ l_h \cdot y_{o_g} \circ \frac{l_p}{l_h}.$$

So we have $l_f = \frac{\partial l_g}{\partial y_{o_g}} \circ l_h \cdot y_{o_g} \circ l_p$ and hence

$$l_p = y_{-o_g} \circ \left(l_f \Big/ (\frac{\partial l_g}{\partial y_{o_g}} \circ l_h) \right) \tag{14}$$

(2). rank(p)<rank(h). By Lemma 14, the leading term of $\sum_{0 \le j \le o_g} \frac{\partial q}{\partial y_j} \circ h \cdot y_j \circ p$ is

$$\frac{\partial q}{\partial y_n} \circ l_h \cdot y_n \circ l_p = b_n q \circ l_h \cdot y_n \circ \frac{l_p}{l_h}.$$

If $n \ge k$, by rank(l_p)<rank(l_h), we have rank($y_k \circ \frac{l_p}{l_h}$)>rank($y_n \circ \frac{l_p}{l_h}$) and the rank of $a_k l_g \circ l_h \cdot y_k \circ \frac{l_p}{l_h}$ is higher than that of $b_i q \circ l_h \cdot y_n \circ \frac{l_p}{l_h}$. Then the leading term of $\sum_{0 \le j \le o_g} \frac{\partial g}{\partial y_j} \circ h \cdot y_j \circ p$ is

$$\frac{\partial l_g}{\partial y_k} \circ l_h \cdot y_k \circ l_p = a_k l_g \circ l_h \cdot y_k \circ \frac{l_p}{l_h}.$$

So we have $l_f = \frac{\partial l_g}{\partial y_k} \circ l_h \cdot y_k \circ l_p$ and hence

$$l_p = y_{-k} \circ \left(l_f \Big/ (\frac{\partial l_g}{\partial y_k} \circ l_h) \right). \tag{15}$$

If $n < k$, it is easy to check that rank$(\frac{\partial l_g}{\partial y_k})$>rank$(\frac{\partial q}{\partial y_n})$ and rank$(y_k \circ l_p)$>rank$(y_n \circ l_p)$. So the rank of $\frac{\partial l_g}{\partial y_k} \circ l_h \cdot y_k \circ l_p$ is higher than that of $\frac{\partial q}{\partial y_n} \circ l_h \cdot y_n \circ l_p$ and the equality (15) still holds.

Since tdeg$(p) <$ tdeg(h) and p, h are both homogeneous, the case of rank(p)=rank(h) will not happen. Hence l_p can be computed from (14) and (15). Let $p = l_p + \bar{p}$. Since $y_j \circ p = y_j \circ l_p + y_j \circ \bar{p}$, $\bar{p}$ still satisfies a formula similar to (13). Therefore, p can be computed by using (14) and (15) repeatedly.

If there exist two distinct difference polynomials p_1 and p_2 such that

$$\begin{cases} f = \sum\limits_{0 \le j \le o_g} \frac{\partial g}{\partial y_j} \circ h \cdot y_j \circ p_1 \\ f = \sum\limits_{0 \le j \le o_g} \frac{\partial g}{\partial y_j} \circ h \cdot y_j \circ p_2 \end{cases}$$

then we have

$$\sum_{0 \le j \le o_g} \frac{\partial g}{\partial y_j} \circ h \cdot y_j \circ (p_1 - p_2) = 0.$$

$p_1 - p_2$ still satisfies the conditions of the lemma. Then either (14) or (15) must hold. We have $l_{p_1 - p_2} = 0$, which means $p_1 = p_2$, a contradiction. ∎

Following the proof of Lemma 15, we give the following algorithm.

Algorithm 5. Input: homogeneous difference polynomials f, g, h.

Output: a difference polynomial p such that $f = \sum\limits_{0 \le i \le o_g} \frac{\partial g}{\partial y_i} \circ h \cdot y_i \circ p$ and tdeg$(p) <$ tdeg(h), if it exists.

S1 Let l_f, l_g, l_h be the leading terms of f, g, h respectively, $k = \min\{i : y_i \text{ appears in } l_g\}$ and $p := 0$. By the proof of Lemma 15, the leading term l_p of p must satisfy one and only one of the following two equalities

$$\begin{cases} \frac{\partial l_g}{\partial y_k} \circ l_h \cdot y_k \circ l_p = l_f & \text{if rank}(p)\text{>rank}(h) \\ \frac{\partial l_g}{\partial y_{o_g}} \circ l_h \cdot y_{o_g} \circ l_p = l_f & \text{if rank}(p)\text{<rank}(h) \end{cases}$$

S2 If $f = 0$, then return p. Use (15) to compute l_p. If l_p is not a difference polynomial, then go to S3; otherwise go to S4. This step computes l_p in the case of rank(p)>rank(h).

S3 Use (14) to compute l_p. If l_p is not a difference polynomial, then terminate the algorithm and return "p does not exist"; otherwise go to step S4. This step computes l_p in the case of rank(p)<rank(h).

S4 Let $p := p + l_p$, $f := f - \sum\limits_{0 \le j \le o_g} \frac{\partial g}{\partial y_j} \circ h \cdot y_j \circ l_p$. Go to step S2.

Now we give the proof for Theorem 13.

If f has a decomposition $f = g \circ h$, we have

$$F_t + F_{t-1} + \cdots + F_1 = (G_a + G_{a-1} + \cdots + G_1) \circ (H_b + H_{b-1} + \cdots + H_1).$$

Comparing the homogeneous parts on both sides of $f = g \circ h$, we have

$$
\begin{aligned}
F_t &= G_a \circ H_b \\
F_{t-1} &= \sum_{0\le i\le o_{G_a}} \frac{\partial G_a}{\partial y_i} \circ H_b \cdot y_i \circ H_{b-1} \\
&\cdots \\
F_{t-k} &= \boxed{G_a \circ (H_b + \cdots + H_{b-k+1})}_{t-k} + \sum_{0\le i\le o_{G_a}} \frac{\partial G_a}{\partial y_i} \circ H_b \cdot y_i \circ H_{b-k} \\
&\cdots \\
F_{t-b+1} &= \boxed{G_a \circ (H_b + H_{b-1} + \cdots + H_2)}_{t-b+1} + \sum_{0\le i\le o_{G_a}} \frac{\partial G_a}{\partial y_i} \circ H_b \cdot y_i \circ H_1
\end{aligned}
\tag{16}
$$

where $\boxed{u}_i$ is the sum of terms in u with total degree i. Using Algorithm 4, we can determine all possible G_a and H_b from the first formula in (16). By Lemma 15, we can determine $H_{b-1}, \cdots, H_1$ uniquely one by one starting from the second equality. So we have $h = H_b + H_{b-1} + \cdots + H_1$. Such an h is not necessarily a right decomposition factor of f, because we have only compared the terms with total degrees from $t-b+1$ to t. We need to check whether h is a right decomposition factor of f, which can be done by Algorithm 1. ∎

Based on the analysis in this section, we have the following decomposition algorithm.

Algorithm 6. Input: a difference polynomial f.
Output: a nontrivial decomposition of f: $f = g \circ h$.

S1 Write f as the sum of homogeneous parts $f = F_t + F_{t-1} + \cdots + F_1$.
S2 Execute Algorithm 4 with input F_t. Assume that the output is T.
S3 If T is empty, then return "f has no nontrivial decompositions"; otherwise, choose a $(g, h) \in T$ and $T := T - \{(g, h)\}$.
S4 Let $a = \mathrm{tdeg}(g), b = \mathrm{tdeg}(h)$, $G_a = g$, and $H_b = h$. For $k = 1, \ldots, b-1$, execute Algorithm 5 to compute $H_{b-k}, \ldots, H_1$ with input

$$F_{t-k} - \boxed{G_a \circ (H_b + H_{b-1} + \cdots + H_{b-k+1})}_{t-k}, G_a, H_b$$

respectively.
S5 Let $h = H_b + H_{b-1} + \cdots + H_1$ and compute a left decomposition factor g such that $f = g \circ h$ by Algorithm 1. If g exists, return (g, h); otherwise, go to S3.

The worst case complexity of the algorithm is exponential. The reason is due to the combinatorial selection in several places. For instance, in step S5 of Algorithm 2, we need to consider all the decomposition factors of a linear difference polynomial, which could be exponential.

Despite of the exponential complexity, the algorithm is quite effective practically. There two reasons for this. First, all the algorithms use explicit formulas to compute the result and as a consequence, for each of the combinatorial selection, the computation is extremely fast. Second, in most cases, the number of possible the combinatorial selections is not very large. The experimental results given in the next section support this assertion.

A consequence of the algorithm is that we do not need to introduce new parameters in the decomposition process.

Example 16. Let $f = (x+1)y_1y_2^3y_3^2 + xyy_1^3y_2^2 + (x+1)(x+2)y_1y_2^2y_3 + (x+1)^2y_2^2y_3^2 + x(x+1)yy_1^2y_2 + x^2y_1^2y_2^2 + y_1y_2^2 + xyy_1^2 + (x+1)^2(x+2)y_2y_3 + x^2(x+2)y_1y_2 + (x+1)y_2 + x^2y_1$.

In step S1, write f as the sum of homogeneous parts $f = F_6 + F_4 + F_3 + F_2 + F_1$, where $F_6 = (x+1)y_1y_2^3y_3^2 + xyy_1^3y_2^2$, $F_4 = (x+1)(x+2)y_1y_2^2y_3 + (x+1)^2y_2^2y_3^2 + x(x+1)yy_1^2y_2 + x^2y_1^2y_2^2$, $F_3 = y_1y_2^2 + xyy_1^2$, $F_2 = (x+1)^2(x+2)y_2y_3 + x^2(x+2)y_1y_2$, $F_1 = (x+1)y_2 + x^2y_1$.

In step S2, using Algorithm 4, we obtain the decomposition of F_6: $T = \{(x+1)y_1y_2 + xyy_1, yy_1^2), (y_1 + y, xyy_1^3y_2^2)\}$ (see Example 12).

In step S3, choose $\big((x+1)y_1y_2 + xyy_1, yy_1^2\big)$ and in step S4, we obtain $H_3 = yy_1^2, H_2 = 0, H_1 = xy_1$. In step S5, we can check that $h = H_3 + H_2 + H_1 = yy_1^2 + xy_1$ is a right decomposition factor of f: $f = \big((x+1)y_1y_2 + xyy_1 + y_1 + xy\big) \circ (yy_1^2 + xy_1)$.

In the following example, we will show an application of decomposition algorithm in solving difference equations.

Example 17. Find the sequence $\{x_n\}$ determined by the recursive equation: $x_{n+2} = x_{n+1}^2 - (2x_n - 1)x_{n+1} + x_n^2$ and $x_0 = 0, x_1 = 2$.

Let $y = y(n) = x_n$ be the function of the discrete variable n and $y_i = y(n+i)$ the i-th transform of y. Then the recursive relation is equivalent to the difference equation $f = y_2 - y_1^2 + (2y-1)y_1 - y^2 = 0$. By Algorithm 6, we can decompose f as $f = (y_1 - y^2) \circ (y_1 - y)$. So the equation is split into two "simpler" ones $\begin{cases} z_1 - z^2 = 0 \\ y_1 - y = z \end{cases}$, where z is also a function of n and z_1 denotes the difference operation on z. The general solution of the first equation is $z = c_1^{2^n}$ and by substituting z into the second equation we have $y(n) = \sum\limits_{0\le k\le n-1} c_1^{2^k} + c_2$, where c_1, c_2 are difference constants. By the initial conditions, we have $c_1 = 2,\ c_2 = 0$. So the sequence $\{x_n\}$ is $x_n = \sum\limits_{0\le k\le n-1} 2^{2^k}$.

6. Compute Decomposition Factors with Parameters

In Example 2, we showed that the decomposition factors of a linear difference polynomial may contain parameters. Fortunately, these parameters can be handled with algebraic equations as shown by the following result.

Proposition 18. *Let f be a linear difference polynomial of order n. Then each right decomposition factor g of f with order m can be written as a difference polynomial with coefficients in $\mathbb{F} = C_{\mathcal{K}}[c_1, \ldots, c_N]$, where $C_{\mathcal{K}}$ is the invariant subfield of $\mathcal{K}$ under δ, $N \le (m+1)\cdot \binom{n}{m}$, and c_i are constant parameters satisfying a set of algebraic equations $e_i = 0, i = 1, \ldots, s$ for $e_i \in \mathbb{F}$.*

Proof. Note that to find a right decomposition factor of a linear difference polynomial is equivalent to find a right factor of a linear difference operator. By the algorithm proposed in (1), to compute a right factor with order m of a given difference operator with order n, we need to compute the hyperexponential solutions of $m+1$ linear homogeneous difference equations with order not greater than $\binom{n}{m}$. By Theorem XII in page 272 of (6), we can conclude that the number of the parameters which may appear in the right factor is not greater than $(m+1)\cdot\binom{n}{m}$. Furthermore, the parameters take values in the invariant subfield of $\mathcal{K}$ under δ and they satisfy some algebraic equations. ▮

In all the algorithms given in Sections 2, 3, 4, and 5, we assume that the parameters do not exist. If the parameters occur, we need to modify some of the algorithms to treat these parameters. We say that a computation step or an algorithm is *affected by the parameters* if distinct parametric values may result in different results in this step or algorithm. For instance, if f and g are difference polynomials with parameters, then $f+g$ is not affected by the parameters, but l_f is affected by the parameters since some parametric values may vanish the leading coefficient of f and hence change l_f. Let us first check which algorithms are affected by the existence of parameters.

Consider the main Algorithm 6. Step S1 is not affected by parameters. Step S2 is affected by the parameters. In Step S2, we call Algorithm 4. The input difference polynomial of Algorithm 4 is from the main algorithm and does not contain parameters. The computation steps in this algorithm do not involve parameters either. The output of this algorithm contains parameters which are introduced when executing Algorithm 2.

The input difference polynomial f of Algorithm 2 is from Algorithm 4 and does not contain parameters. The computation in this algorithm does not involve parameters. The parameters are introduced in Step S5 when decomposing linear difference polynomials.

In Algorithm 2, we need to call Algorithm 3. The input difference polynomials f and g of Algorithm 3 contain parameters. The only step affected by the parameters is Step S2, where we need to compute an inversion y_{-o_g} and a quotient i_f/i_g which are affected by the parameters.

Now we have checked Step S2 of the main algorithm. Step S3 does not affected by the parameters. Step S4, which is Algorithm 5, is affected by the parameters. The input difference polynomials f and h of Algorithm 5 may contain parameters. In step S1 of Algorithm 5, we need to compute the leading terms of f, g, h, which is affected by the parameters. In Steps S2 and S3, we use (15) and (14) which are also affected by the parameters. Here again, we need to compute an inversion and a quotient.

Now return to the main algorithm. Step S5 is basically Algorithm 1 which is affected by the parameters. The input difference polynomials of Algorithm 1 may contain parameters. In Steps S1 and S2, we need to compute o_f, d_f, d_h, and o_h which are affected by the parameters. We need to compute the leading terms of f, g, h, which is also affected by the parameters. In Step S3, we need to compute a quotient which is affected by the parameters.

Summarizing the above analysis, we have the following conclusion. The computations affected by the parameters include three categories:

(1) Compute o_f, d_f, l_f for a difference polynomial f containing parameters.
(2) Compute an inversion $y_{-o}\circ f$ for a difference polynomial f containing parameters.
(2) Compute the quotient f/g for two difference polynomials f and g containing parameters.

As a consequence, we need only show how to treat these three computation problems.

To make the problem precise, we introduce the following notations. Let $c_1,\dots,c_N$ be some constant parameters, $\mathbb{P}=\{p_1,\dots,p_s\}$, and $\mathbb{D}=\{d_1,\dots,d_t\}$ where $p_i, d_j\in\mathbb{F}=\mathcal{C}_{\mathcal{K}}[c_1,\dots,c_N]$ and $\mathcal{C}_{\mathcal{K}}$ is the constant field of $\mathcal{K}$ under the transforming operator. Let $\mathcal{E}$ be an algebraically closed extension field of $\mathcal{C}_{\mathcal{K}}$. We use $\mathrm{Zero}(\mathbb{P}/\mathbb{D})$ to denote all $\eta\in\mathcal{E}^N$ such that $p_i(\eta)=0, i=1,\dots,s$ and $d_j(\eta)\neq 0, j=1,\dots,t$.

By Proposition 18, we may assume that a difference polynomial f containing parameters is represented by

$$f \in \mathbb{F}\{y\} \text{ and } \mathrm{Zero}(\mathbb{P}/\mathbb{D})$$

meaning that the coefficients of f are constrained by $\mathbb{P} = \{p_1 = 0, \ldots, p_s = 0\}$ and $\mathbb{D} = \{d_1 \neq 0, \ldots, d_t \neq 0\}$.

First, consider the computation of l_f. Let $l_f = C_f M_f$ where $M_f = \prod_{i=0}^{o_f} y_i^{n_i}$ and $C_f \in \mathcal{K}[c_1, \ldots, c_N]$. It is clear that

$$\mathrm{Zero}(\mathbb{P}/\mathbb{D}) = \mathrm{Zero}(\mathbb{P}/\mathbb{D} \cup \{C_f\}) \cup \mathrm{Zero}(\mathbb{P} \cup \{C_f\}/\mathbb{D}).$$

Check whether $C_1 = \mathrm{Zero}(\mathbb{P}/\mathbb{D} \cup \{C_f\}) = \emptyset$ and $C_2 = \mathrm{Zero}(\mathbb{P} \cup \{C_f\}/\mathbb{D}) = \emptyset$ with the characteristic set method (22) or the Gröbner basis method (3).

If $C_1 \neq \emptyset$, then C_f is not zero for any element of C_1, that is, $l_f = C_f M_f$ is always valid on C_1. In this case, we can continue our algorithm on C_1. If $C_1 = \emptyset$, then C_f is identically zero on C_1. Note that in this case, we have $C_2 = \mathrm{Zero}(\mathbb{P}/\mathbb{D})$ and $l_f = l_{f - C_f M_f}$ on C_2. We may repeat the above procedure for $f_1 = f - C_f M_f$ on C_2.

Intuitively speaking, the above procedure is to divide the parametric space into disjoint regions such that on each region, l_f is well defined. We can compute o_f, d_f, and $y_{-o}f$ in a similar way.

Let f and g be difference polynomials whose coefficients satisfy the constraints $\mathbb{P} = \{p_1 = 0, \ldots, p_s = 0\}$ and $\mathbb{D} = \{d_1 \neq 0, \ldots, d_t \neq 0\}$ for $p_i, d_j \in \mathbb{F}$. We now show how to compute the quotient $q = f/g$. We always have $o_q \leq o_f$ and $\mathrm{tdeg}(q) \leq \mathrm{tdeg}(f)$. To compute a q such that $f = q \cdot g$, let $q = \sum_I g_I y^{i_0} y_1^{i_1} \cdots y_c^{i_c}$, where $c = o_f$ and $i_0 + i_1 + \cdots + i_c \leq \mathrm{tdeg}(f)$. Substituting f, g, q into $f = q \cdot g$ and comparing the coefficients of the monomials on both sides, we obtain a parametric linear system $\mathbb{L}$ about the coefficients g_I. Solve the parametric system $\mathbb{P} = 0, \mathbb{L} = 0$ and $\mathbb{D} \neq 0$ with methods from (8; 18), we may obtain a decomposition:

$$\mathrm{Zero}(\mathbb{P} \cup \mathbb{L}/\mathbb{D}) = \cup_k \mathrm{Zero}(\mathbb{P}_k/\mathbb{D}_k)$$

such that on each component $C_k = \mathrm{Zero}(\mathbb{P}_k/\mathbb{D}_k)$ we have a difference polynomial q_k whose leading term is not zero on C_k and $f = q_k \cdot g$. In the rest of the computation, we need to consider each of the quotient q_k on the component C_k.

We may give a parametric version for Algorithm 1 according to the above idea. If $f = g \circ h$, we always have $o_g \leq o_f$ and $\mathrm{tdeg}(g) \leq \mathrm{tdeg}(f)$. To compute a g such that $f = g \circ h$, let $g = \sum_I g_I y^{i_0} y_1^{i_1} \cdots y_c^{i_c}$, where $c = o_f$ and $i_0 + i_1 + \cdots + i_c \leq \mathrm{tdeg}(f)$. Substituting f, g, h into $f = g \circ h$ and comparing the coefficients of the monomials on both sides, we obtain a parametric linear system $\mathbb{L}$ about g_I. The other steps are similar.

7. Experimental Results

We implement Algorithm 6 in Maple for the constant field case $\mathcal{K} = \mathbb{Q}$. The reason is that if $f \in \mathbb{Q}\{y\}$ and is linear then computing the decomposition factors of f is equivalent to the factorization of a univariate polynomial with coefficients in $\mathbb{Q}$. In this case, there exist no parameters. To implement Algorithm 6 for $\mathcal{K} = \mathbb{Q}(x)$, we need an implementation which could give all the possible factorizations of a linear difference operator, which is not available in Maple. Besides this point, our implementation suits the case of $\mathcal{K} = \mathbb{Q}(x)$.

Two sets of experiments are done. In Table 1, we generate a difference polynomial randomly and decompose it. All the randomly generated difference polynomials in Table 1 are indecomposable. In Table 1, o_f and t_f are the order and the total degree of the tested difference polynomials respectively. For each value of (o_f, t_f), we run ten examples. The index n_f is the average number of the terms in the ten difference polynomials. The column starting with time(s) is the average running time for the ten examples in seconds. The running times are collected on a PC with a 3.2GHz CPU and 512M of memory.

(o_f, t_f, n_f)	time(s)	(o_f, t_f, n_f)	time(s)
(2,15, 399.6)	0.0546	(2,20,1067.6)	0.220
(2, 30,2612.4)	0.986	(2, 35,3250.2)	1.301
(3,10,538.3)	0.1376	(3, 15, 1735.3)	0.4313
(3,20,3431.3)	1.976	(3, 25, 14140.8)	6.984
(4,10,1275.4)	0.465	(4, 15, 8809.7)	8.094
(5,10,4332.6)	2.646	(5, 12, 9290.2)	11.15
(6,8,3064.2)	1.914	(6,11,14985)	13.737
(7,8,6919.2)	7.118	(7,9,13212.3)	21.44
(8,6,2849.4)	3.196	(8,8,16826.9)	34.965

Table 1. Decomposing Randomly Generated Difference Polynomials

In Table 2, we generate two difference polynomials g and h randomly and decompose $f = g \circ h$. The difference polynomials g and h are given in Table 3.

From these experimental results, we may conclude that our algorithm is efficient in handling large difference polynomials with thousands of terms. The computational efficiency is due to the fact that all computations in the algorithm are based on explicit

g, h	(o_g, t_g)	(o_h, t_h)	(o_f, t_f, n_f)	time(s)
g_1, h_1	(0,8)	(1,8)	(1,64,761)	6.671
g_2, h_2	(1,6)	(1,8)	(2,48,2302)	1.969
g_3, h_3	(1,6)	(2,4)	(3,24,1538)	1.375
g_4, h_4	(1,4)	(3,4)	(4,16,2329)	0.547
g_5, h_5	(2,4)	(1,6)	(3,24,1558)	0.437
g_6, h_6	(2,4)	(2,6)	(4,24,363)	3.782
g_7, h_7	(2,4)	(3,4)	(5,16,716)	0.672
g_8, h_8	(3,4)	(2,6)	(5,24,4475)	2.141
g_9, h_9	(3,4)	(3,4)	(6,16,10079)	9.984
g_{10}, h_{10}	(4,3)	(3,4)	(7,12,807)	0.766

Table 2. Decomposing $f = g \circ h$ for randomly generated g and h

g_1	$-37y^8 - 20y^7 - 29y^6 + 48y^5 + 20y^4 - 3y^3$
h_1	$48y - 18yy_1 - 48y^4 + 42y^3y_1^2 + 2y^6 + 41y^4y_1^3 + 19y^4y_1^4$
g_2	$-10y^2y_1 - 11y_1^3 - 31y^5y_1$
h_2	$32y + 19y_1 - 39y^3 - 19y^2y_1 - 24y^2y_1^2 - y^5y_1^2 - 9y^3y_1^5 + 29yy_1^7$
g_3	$13y^2 - 49y^2y_1^2 - 21y^3y_1^2 + 17y_1^5 + 21y^4y_1^2$
h_3	$38y_1y_2^2 - 13y^3y_1 - 37yy_1^3 + 19y_1^3y_2 - 40y_1y_2^3$
g_4	$36y - 50y_1 - 17y^2y_1 + 10y_1^2y - 21y_1^3 - 22y_1^2y^2 - 3y_1^3y$
h_4	$48y_1^2 - 32y^2y_1 - 40y_2^2y_3 + 23yy_2^2y_3 + yy_3^3 + 8y_1^2y_2y_3 + 5y_1^2y_3^2 + 32y_2^4$
g_5	$-5y + 44y_1^3 + 16y_1^2y_2 + 14y^2y_2^2 - 13yy_1y_2^2 + 31y_1^2y_2^2$
h_5	$-18yy_1 + 15y^3 - 20y_1^4 - 3yy_1^4 + 28y^6 - 19y^4y_1^2$
g_6	$17y^2 + 20y_1y_2 - 18y^2y_1 + 47y^2y_2 + 43y^2y_2^2 - 46yy_1^2y_2$
h_6	$43y^3 - 34y_1^4 + 25y^4y_1 - 47y^2y_1^3y_2$
g_7	$35y_2 - 30y^3 - 19y_1^3 - 18y^3y_1 + 2yy_1^2y_2 + 19y_1y_2^3$
h_7	$3y^2y_1 + 46y^2y_2^2 + 13yy_1^3 - 5y_1y_2^2y_3 + 26y_1y_3^3$
g_8	$-5y - 9y_3y^2 + 28y_2^2y - 45y_1y_2^2 + 32y_1y^3 - 24y_2y^3 - 16y_2^2y^2$ $+28y_2^2y_3y - 16y_1^2y_3^2 + 7y_2y_3^3$
h_8	$45yy_2 - 41y_1^3 - 6y_1^5 - 44y_1^3y_2^2 + 16y_1^2y_2^3 + 6y_1y_2^4 + 32y_1y_2^2y^3$
g_9	$-41y^2 - 10y_1y_2^2 + 25y^2y_1^2 - 29yy_1y_3^2 + 5yy_2y_3^2 - 48y_2^2y_3^2$
h_9	$18y^2 + 36y_2^2 - 37y^3y_2 - 3y^2y_1y_3 + 20y^2y_2^2 + 33y^2y_2y_3 + 46yy_1^3$ $+31yy_1^2y_3 + 37y_1^3y_2 - 5y_1^2y_2y_3$
g_{10}	$21y_2 - 13y_3 + 6yy_1 - 23y_3y_4 + 12yy_1y_3 - 38yy_1y_4 - 6yy_2^2$ $-16yy_4^2 - 24y_1^2y_3 + 18y_2^3$
h_{10}	$-26yy_1 + 14y_2y_3 + 32y^2y_2 + 29y^2y_3 - 27y_1^2y_3 - 46y_2^2y_3^2$

Table 3. Randomly generated g and h

formulas. Our program is especially fast for randomly generated difference polynomials. The reason can be explained below. From Lemmas 10 and 15, we can see that a difference polynomial with a nontrivial decomposition has certain structures and a randomly generated difference polynomial does not have these structures. As a consequence, the program will stop early before going through all the cases.

Based on these experimental results, we may conclude that our algorithm provides an efficient reduction of the decomposition of nonlinear difference polynomials to the linear case.

8. Conclusions

In this paper, we give a complete algorithm to decompose a given nonlinear ordinary difference polynomial. The algorithm provides an efficient reduction of the problem to the decomposition of linear difference polynomials. The algorithm is implemented and can be used to decompose difference polynomials with thousands of terms. The treatment of the parametric case in Section 6 is not satisfactory due to the occurrence of general polynomial equation systems. It seems that this problem should be studied by first exploring the detailed structure of the factors of linear difference operators.

Besides the algorithmic study for the decomposition, the uniqueness problem is also an important property to be explored. Ritt gave a perfect result for the uniqueness of decompositions for an algebraic polynomial (15). Similar results were proved for Ore polynomials and hence for linear difference polynomials (13). It is interesting to see whether these properties can be extended to the case of nonlinear difference polynomials.

References

[1] M. Bronstein and M. Petkovšek, An Introduction to Pseudo-linear Algebra, *Theoretical Computer Science*, **157**, 3-33, 1996.

[2] M. Bronstein and M. Petkovšek, On Ore Rings, Linear Operators and Factorization, *Programmirovanie*, **20**, 27-45, 1994.

[3] B. Buchberger, Gröbner Bases: An Algorithmic Method in Polynomial Ideal Theory. *Recent Trends in Multidimensional Systems theory* (ed. N.K. Bose), D.Reidel Publ. Comp., 1985.

[4] D. Kozen and S. Landau, Polynomial decomposition algorithms. *Journal of Symbolic computation 7*, 445-456, 1989.

[5] J. von zur Gathen, Functional Decomposition of Polynomials: The Wild Case. *Journal of Symbolic Computation*, **9**, 437-452, 1990; The Tame Case. *Journal of Symbolic Computation*, **9**, 281-299, 1990.

[6] R.M. Cohn, *Difference Algebra*, Interscience Pbulishers, 1965.

[7] D.A. Cox, J.B. Little, and D. O'Shea, *Ideals, Varieties, and Algorithms*, Springer, Berlin, 1997.

[8] X.S. Gao and S.C. Chou, Solving Parametric Algebraic Systems, *Proc. of ISSAC'92*, 335-341, ACM Press, New York, 1992.

[9] X.S. Gao and M. Zhang, Decomposition of Differential Polynomials with Constant Coefficients, *Proc. ISSAC'04*, 175-182, ACM Press, New York, 2004.

[10] X.S. Gao and M. Zhang, Decomposition of Ordinary Differential Polynomials, *Applicable Algebra in Engineering, Communication and Computing*, **19**(1), 1-25, 2008.

[11] M. Giesbrecht and Y. Zhang, Factoring and Decomposing Ore Polynomials Over $F_q(t)$, *Proc. ISSAC'03*, 127-134, ACM Press, New York, 2003.

[12] Z. Li, A Subresultant Theory for Ore Polynomials with Applications, *Proc. ISSAC'98*, 132 - 139, ACM Press, New York, 1998.

[13] O. Ore, Theory of Noncommutative Polynomials. *Annals of Mathematics*, **34**(3), 480-508, 1933.

[14] J.F. Ritt, *Differential Algebra*, AMS, New York, 1950.

[15] J.F. Ritt, Prime and Composite Polynomials, *Trans. Amer. Math. Soc.*, **23**, 51-66, 1922.

[16] A. Schinzel, *Polynomials with Special Regard to Reducibility*, Cambridge University Press, 2000.

[17] M.F. Singer, Liouillian Solutions of nth order Homogeneous Linear Differential Equations. *Amer. J. of Math.*, 103(4), 661-682, 1981.

[18] W.Y. Sit, An Algorithm for Solving Parametric Linear Systems, *Journal of Symbolic Computation*, **13**, 353-394, 1992.

[19] M.V. Sosnin, An Algorithm for Nonparametric Decomposition of Differential Polynomials, *Programming and Computing Software*, **27**(1), 43 - 49, 2001.

[20] S.P. Tsarev, On Factorization of Non-linear Ordinary Differential Equations, *Proc. ISSAC'99*, 159-164, ACM Press, New York, 1999.

[21] M. Van der Put and M.F. Singer, *Galois Theory of Linear Differential Equations*, Springer, Berlin, 2003.

[22] W.T. Wu, *Basic Principles of Mechanical Theorem Proving in Geometries.* Springer, Wien, 1995.