MATHEMATICS OF COMPUTATION
Volume 74, Number 250, Pages 853–868
S 0025-5718(04)01710-7
Article electronically published on August 20, 2004

ON SOLVING COMPOSITE POWER POLYNOMIAL EQUATIONS

YINGQUAN WU AND CHRISTOFOROS N. HADJICOSTIS

Abstract. It is well known that a system of power polynomial equations can be reduced to a single-variable polynomial equation by exploiting the so-called *Newton's identities*. In this work, by further exploring Newton's identities, we discover a binomial decomposition rule for composite elementary symmetric polynomials. Utilizing this decomposition rule, we solve three types of systems of *composite* power polynomial equations by converting each type to single-variable polynomial equations that can be solved easily. For each type of system, we discuss potential applications and characterize the number of nontrivial solutions (up to permutations) and the complexity of our proposed algorithmic solution.

1. Introduction

Systems of algebraic equations appear in many application areas of computational algebra, including communications, robotics, chemistry, and mechanics. In these algebraic equations, one commonly aims at carrying out eliminations in order to obtain a triangular system of equations which can be easily solved. The most widely known such methods include resultants, Gröebner bases, and characteristic sets. On the other hand, a system of power polynomial equations, i.e., a system of the form

$$\sum_{i=1}^{n} x_i^j = s_j, \quad j = 1, 2, \ldots, n, \tag{1.1}$$

can be solved by reducing it to a single-variable polynomial equation using Newton's identities, which are based on simple relationships between elementary symmetric polynomials and power polynomials [1].

In this work, by further analyzing Newton's identities, we discover a binomial decomposition rule for composite elementary symmetric polynomials. We apply this decomposition rule towards the solution of three types of systems of composite power polynomial equations, each time reducing the system to a single-variable polynomial equation that can be solved efficiently using standard techniques.

Received by the editor October 9, 2002 and, in revised form, September 5, 2003.
2000 *Mathematics Subject Classification.* Primary 65H10; Secondary 12Y05.
Key words and phrases. Power polynomial, composite power polynomial, Newton's identities, system of polynomial equations.
This material is based upon work supported in part by the National Science Foundation under NSF Career Award 0092696 and NSF ITR Award 0085917 and in part by the Motorola Research Center at the University of Illinois. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the NSF or Motorola.

(i) *Type A polynomial system:* The first system of composite power polynomial equations that we consider is of the form

$$\sum_{i=1}^{k} x_i^j - \sum_{i=k+1}^{n} x_i^j = s_j, \quad j = 1, 2, \ldots, n \,. \tag{1.2}$$

This system is shown to have a unique nontrivial solution (upon permutation of the parameters $x_1, x_2, \ldots, x_n$). A solution is considered trivial if it satisfies $x_i = x_j \neq 0$, $1 \leq i \leq k$, $k+1 \leq j \leq n$.

(ii) *Type B polynomial system:* A Type B polynomial system is a natural extension of a Type A polynomial system and takes the form

$$\sum_{i=1}^{n} c_i x_i^j = s_j, \quad j = 1, 2, \ldots, n, \quad c_i \in \{0, \pm 1\} \,. \tag{1.3}$$

This system has $2n$ unknowns coupled with n equations and is shown to have up to $n+1$ distinct nontrivial solutions (upon permutations).

(iii) *Type C polynomial system:* A Type C polynomial system is in the form

$$\sum_{i=1}^{n-1} x_i^j - \alpha x_n^j = s_j, \quad j = 1, 2, \ldots, n,$$

where α is an arbitrary number.

Before we discuss properties and algorithms for solving the three types of polynomial systems mentioned above, we briefly discuss their applicability. Due to the existence of the efficient solving algorithm that we present in this paper, Type A polynomial systems have been employed towards the design of fault identification schemes in discrete event systems that can be modeled by Petri nets [2]. More specifically, this work considers two types of Petri net transition faults, post-condition and pre-condition faults, and, by adding d redundant places to the given Petri net, it reduces the problem of identifying transition faults to a system of equations of the form $\mathbf{De} = \mathbf{s}$, where the d-dimensional vector $\mathbf{s}$ is a known *syndrome* and $\mathbf{e}$ is the desired indicator vector of transition fault events (with a "+1" entry corresponding to a post-condition fault and a "−1" entry corresponding to a pre-condition fault). Therefore, by choosing matrix $\mathbf{D}$ to be a Vandermonde matrix, the problem of transition fault identification is reduced to a Type A polynomial system that can be solved efficiently [2].

A Type B polynomial system can be viewed as a variation of algebraic decoding of Reed-Solomon codes [3]. The decoding of a Reed-Solomon code involves solving for the unique solution of a system with n unknowns coupled with n equations, described (for even n) by

$$\sum_{i=1}^{n/2} c_i x_i^j = s_j, \quad j = 1, 2, \ldots, n \,. \tag{1.4}$$

In communications or data storage applications, the coefficients $\{c_i\}$ correspond to the values of the errors (introduced by noise in the system) and the parameters $\{x_i\}$ indicate the location of the errors. Given s_1, s_2,..., s_n, errors are correctable as long as the number of erroneous locations (nonzero c_i's) is smaller than or equal to $n/2$. Notice that a Type B polynomial system restricts the coefficients $\{c_i\}$ to satisfy $c_i \in \{0, \pm 1\}$ but allows for double the number of x_i (which also increases the

potential number of distinct solutions). Thus, solving a Type B polynomial system is desirable when decoding Reed-Solomon codes that are used in applications where errors of the form ± 1 are more likely than arbitrary errors (e.g., when phase-shift-keying modulation is used). Note that errors different from ± 1 can also be handled by considering multiple instances of the same x_i. As an example, consider the case when $n = 6$ and four errors occur, with values $-1, -1, -2, 2$, respectively. Clearly, solving (1.4) will not succeed in identifying the errors because the number of errors (4) is larger than error-correcting capability ($n/2 = 3$). However, a Type B polynomial system is capable of successfully identifying the errors (-2 and 2 are both counted as two faults and thus the number of errors under our model is six; i.e., it is within our capabilities when solving system (1.3)).

2. Preliminaries

In this section we review the necessary background on *power polynomial equations* and *Newton's identities*. *A power polynomial* is defined as

$$\mathcal{S}_k(x_1, x_2, \ldots, x_n) \triangleq \sum_{i=1}^{n} x_i^k \tag{2.1}$$

and a *system of power polynomial equations* is usually represented as in (1.1). It is well known that the system in (1.1) can be solved by employing *Newton's identities* [1]. The identities display a simple relation between the *elementary symmetric polynomials* given by

$$\Lambda_k(x_1, x_2, \ldots, x_n) \triangleq (-1)^k \sum_{1 \le i_1 < i_2 < \cdots < i_k \le n} x_{i_1} x_{i_2} \cdots x_{i_k}, \quad 1 \le k \le n \tag{2.2}$$

(for consistency, we set $\Lambda_0 = 1$) and the power polynomials $\{\mathcal{S}_i(x_1, x_2, \ldots, x_n)\}_{i=1}^{n}$ in (2.1). More specifically, if we use Λ_k, $\mathcal{S}_k$ to represent $\Lambda_k(x_1, x_2, \ldots, x_n)$, $\mathcal{S}_k(x_1, x_2, \ldots, x_n)$, respectively, Newton's identities take the form

$$\begin{cases} \mathcal{S}_k + \Lambda_1 \mathcal{S}_{k-1} + \cdots + \Lambda_{k-1} \mathcal{S}_1 + k\Lambda_k = 0, & 1 \le k \le n, \\ \mathcal{S}_k + \Lambda_1 \mathcal{S}_{k-1} + \cdots + \Lambda_{n-1} \mathcal{S}_{k-n+1} + \Lambda_n \mathcal{S}_{k-n} = 0, & k > n, \end{cases} \tag{2.3}$$

and *remain true over an arbitrary field* [3]. If $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n$ are known and $\frac{1}{n!}$ is defined in the given field, we can use (2.3) to obtain the *linear* equation array

$$\begin{cases} \mathcal{S}_1 + \Lambda_1 & = 0 \\ \mathcal{S}_2 + \Lambda_1 \mathcal{S}_1 + 2\Lambda_2 & = 0 \\ & \vdots \\ \mathcal{S}_n + \Lambda_1 \mathcal{S}_{n-1} + \cdots + \Lambda_{n-1} \mathcal{S}_1 + n\Lambda_n & = 0, \end{cases} \tag{2.4}$$

and we easily solve for Λ_k in terms of $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k$ to obtain $\Lambda_k = \mathcal{F}_k(\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k)$, $k = 1, 2, \ldots, n$. More specifically,

$$\mathcal{F}_k(\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k) = \frac{(-1)^k}{k!} \begin{vmatrix} \mathcal{S}_1 & 1 & 0 & \cdots & 0 & 0 \\ \mathcal{S}_2 & \mathcal{S}_1 & 2 & \cdots & 0 & 0 \\ \mathcal{S}_3 & \mathcal{S}_2 & \mathcal{S}_1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathcal{S}_{k-1} & \mathcal{S}_{k-2} & \mathcal{S}_{k-3} & \cdots & \mathcal{S}_1 & k-1 \\ \mathcal{S}_k & \mathcal{S}_{k-1} & \mathcal{S}_{k-2} & \cdots & \mathcal{S}_2 & \mathcal{S}_1 \end{vmatrix}. \tag{2.5}$$

We remark that the solution for Λ_k in terms of $\{\mathcal{S}_i\}_{i=1}^k$ is independent of the number of original variables n. Moreover, the above expression does *not* hold if the inverse $\frac{1}{k!}$ is not defined in the given finite field. Also note that the equation system (2.4) is a *lower triangular Toeplitz linear* system, whose solution requires $O(n \log n)$ arithmetic operations [4].

Once $\{\Lambda_i(x_1, x_2, \ldots, x_n)\}_{i=1}^n$ are available, the solutions $x_1, x_2, \ldots, x_n$ of (1.1) are exactly the roots of the *characteristic polynomial*

$$x^n + \Lambda_1(x_1, \ldots, x_n)x^{n-1} + \cdots + \Lambda_n(x_1, \ldots, x_n) = \prod_{i=1}^{n}(x - x_i) \tag{2.6}$$

and can be solved using well-established methods.

3. Developments

Note here that even though $\Lambda_k(x_1, x_2, \ldots, x_n)$ is not defined when $k > n$, $\mathcal{F}_k(\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k)$ is. For consistency, in our developments we will extend the definition of $\Lambda_k(x_1, x_2, \ldots, x_n)$ for $k > n$ by setting it equal to $\mathcal{F}_k(\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k)$. In fact, based on this extension, we have

Proposition 1. *Let $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k$ be the power polynomials of $x_1, x_2, \ldots, x_n$. Then,*

$$\Lambda_k(x_1, x_2, \ldots, x_n) \equiv \mathcal{F}_k(\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k) = 0, \quad k > n. \tag{3.1}$$

Proof. We can regard $\{\mathcal{S}_i\}_{i=1}^k$ as power polynomials of $x_1, \ldots, x_n, 0_{n+1}, \ldots, 0_k$, which immediately leads to

$$\mathcal{F}_k(\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k) = \Lambda_k(x_1, \ldots, x_n, 0_{n+1}, \ldots, 0_k) = 0$$

for $k > n$. □

The above extension also enables us to simplify the notation for Newton's identities as follows:

$$\mathcal{S}_k + \Lambda_1\mathcal{S}_{k-1} + \cdots + \Lambda_{k-1}\mathcal{S}_1 + k\Lambda_k = 0, \quad \forall\, k, n \geq 1\ . \tag{3.2}$$

We now use Proposition 1, together with Newton's identities, to analyze the system of equations

$$\begin{cases} x_1 + x_2 + \cdots + x_k &= y_1 + y_2 + \cdots + y_l, \\ x_1^2 + x_2^2 + \cdots + x_k^2 &= y_1^2 + y_2^2 + \cdots + y_l^2, \\ x_1^3 + x_2^3 + \cdots + x_k^3 &= y_1^3 + y_2^3 + \cdots + y_l^3, \\ & \vdots \\ x_1^k + x_2^k + \cdots + x_k^k &= y_1^k + y_2^k + \cdots + y_l^k, \end{cases} \tag{3.3}$$

where $k \geq l$.

Proposition 2. *The system of equations in* (3.3) *has the unique solution $\{x_i\} = \{y_j\}$ when $k = l$ and the unique solution $\{x_i\}_{i=1}^k = \{y_i\}_{i=1}^l \cup \{0\}$ when $k > l$ (up to permutations).*

Proof. For $1 \leq i \leq k$, Newton's identities indicate

$$\begin{aligned} \Lambda_i(x_1, x_2, \ldots, x_k) &= \mathcal{F}_i(\mathcal{S}_1(x_1, \ldots, x_k), \mathcal{S}_2(x_1, \ldots, x_k), \ldots, \mathcal{S}_i(x_1, \ldots, x_k)) \\ &= \mathcal{F}_i(\mathcal{S}_1(y_1, \ldots, y_l), \mathcal{S}_2(y_1, \ldots, y_l), \ldots, \mathcal{S}_i(y_1, \ldots, y_l)) \\ &= \Lambda_i(y_1, y_2, \ldots, y_l). \end{aligned}$$

In particular, Proposition 1 indicates that

$$\Lambda_i(x_1, x_2, \ldots, x_k) = \Lambda_i(y_1, y_2, \ldots, y_l) = 0, \quad l < i \le k.$$

Thus, the characteristic polynomial of $x_1, x_2, \ldots, x_k$ is given by

$$\begin{aligned}\prod_{i=1}^{k}(x - x_i) &= x^k + x^{k-1}\Lambda_1(x_1, x_2, \ldots, x_k) + \cdots + \Lambda_k(x_1, x_2, \ldots, x_k)\\ &= x^k + x^{k-1}\Lambda_1(y_1, y_2, \ldots, y_l) + \cdots + x^{k-l}\Lambda_l(y_1, y_2, \ldots, y_l)\\ &= x^{k-l}\prod_{i=1}^{l}(x - y_i),\end{aligned}$$

which yields the desired conclusion. □

The following theorem indicates another significant property of function $\mathcal{F}$.

Theorem 1. *Let S_i, T_i, $i = 1, 2, \ldots, k$, be $2k$ arbitrary given parameters. Then,*

$$\mathcal{F}_k(S_1 + T_1,\ \ldots,\ S_k + T_k) = \sum_{i=0}^{k}\mathcal{F}_i(S_1,\ \ldots,\ S_i)\mathcal{F}_{k-i}(T_1,\ \ldots,\ T_{k-i}). \tag{3.4}$$

We provide two proofs: the first one only applies in the field of complex numbers but it is intuitive and easy to understand; the second one is more complex but applies to any field where $\frac{1}{k!}$ is defined. Therefore, the decomposition rule in (3.4) is valid in any field where $\frac{1}{k!}$ is defined.

Proof 1. For the first proof, let $y_1, y_2, \ldots, y_k$ be a (possibly complex) solution[1] of the equation array

$$\begin{cases} y_1 + y_2 + \cdots + y_k = T_1,\\ y_1^2 + y_2^2 + \cdots + y_k^2 = T_2,\\ \qquad\qquad \vdots\\ y_1^k + y_2^k + \cdots + y_k^k = T_k, \end{cases}$$

and let $x_1, x_2, \ldots, x_k$ be a (possibly complex) solution of the equation array

$$\begin{cases} x_1 + x_2 + \cdots + x_k = S_1,\\ x_1^2 + x_2^2 + \cdots + x_k^2 = S_2,\\ \qquad\qquad \vdots\\ x_1^k + x_2^k + \cdots + x_k^k = S_k. \end{cases}$$

Then, we have

$$\begin{aligned}\mathcal{F}_k(S_1 + T_1,\ \ldots,\ S_k + T_k) &= \Lambda_k(x_1, \ldots, x_k, y_1, \ldots, y_k)\\ &= \sum_{i=0}^{k}\Lambda_{k-i}(x_1, \ldots, x_k)\cdot\Lambda_i(y_1, \ldots, y_k)\\ &= \sum_{i=0}^{k}\mathcal{F}_{k-i}(S_1, \ldots, S_{k-i})\cdot\mathcal{F}_i(T_1, \ldots, T_i). \quad \square\end{aligned}$$

[1]It is clear that there is always a solution $y_1, y_2, \ldots, y_k$: from $T_1, T_2, \ldots, T_k$, we can find $\Lambda_1, \Lambda_2, \ldots, \Lambda_k$ (through the system of equations (2.3)); then $y_1, y_2, \ldots, y_k$ are the roots of polynomial (2.6).

Proof 2. For the case in which we are operating in an arbitrary finite field, the proof is by induction. For $k = 1$, the conclusion holds trivially. For $k = 2$, the conclusion follows from (2.5):

$$\begin{aligned}
&\mathcal{F}_2(S_1 + T_1, S_2 + T_2) \\
&\quad = \frac{1}{2}\begin{vmatrix} S_1 + T_1 & 1 \\ S_2 + T_2 & S_1 + T_1 \end{vmatrix} \\
&\quad = \frac{1}{2}((S_1 + T_1)^2 - (S_2 + T_2)) \\
&\quad = \frac{1}{2}(S_1^2 - S_2) + S_1T_1 + \frac{1}{2}(T_1^2 - T_2) \\
&\quad = \frac{1}{2}\begin{vmatrix} S_1 & 1 \\ S_2 & S_1 \end{vmatrix} + S_1T_1 + \frac{1}{2}\begin{vmatrix} T_1 & 1 \\ T_2 & T_1 \end{vmatrix}.
\end{aligned}$$

Now we assume the conclusion holds for all $k < n$ and we aim to prove its validity for $k = n$. We first define $\{\Lambda_i^{ST}\}_{i=1}^n$, $\{\Lambda_i^S\}_{i=1}^n$, and $\{\Lambda_i^T\}_{i=1}^n$, such that

$$\begin{cases}
(S_1 + T_1) + \Lambda_1^{ST} & = 0 \\
(S_2 + T_2) + (S_1 + T_1)\Lambda_1^{ST} + 2\Lambda_2^{ST} & = 0 \\
& \vdots \\
(S_n + T_n) + (S_{n-1} + T_{n-1})\Lambda_1^{ST} + \cdots + (S_1 + T_1)\Lambda_{n-1}^{ST} + n\Lambda_n^{ST} & = 0,
\end{cases}$$

$$\begin{cases}
S_1 + \Lambda_1^S & = 0 \\
S_2 + S_1\Lambda_1^S + 2\Lambda_2^S & = 0 \\
& \vdots \\
S_n + S_{n-1}\Lambda_1^S + \cdots + S_1\Lambda_{n-1}^S + n\Lambda_n^S & = 0,
\end{cases}$$

$$\begin{cases}
T_1 + \Lambda_1^T & = 0 \\
T_2 + T_1\Lambda_1^T + 2\Lambda_2^T & = 0 \\
& \vdots \\
T_n + T_{n-1}\Lambda_1^T + \cdots + T_1\Lambda_{n-1}^T + n\Lambda_n^T & = 0.
\end{cases}$$

Formally,

$$\Lambda_i^{ST} = \mathcal{F}_i(S_1 + T_1, S_2 + T_2, \ldots, S_i + T_i),$$
$$\Lambda_i^S = \mathcal{F}_i(S_1, S_2, \ldots, S_i),$$

and

$$\Lambda_i^T = \mathcal{F}_i(T_1, T_2, \ldots, T_i)$$

for $i = 1, 2, \ldots, n$. Therefore, it is equivalent to proving that

$$\Lambda_n^{ST} = \sum_{i=0}^{n} \Lambda_i^S \Lambda_{n-i}^T,$$

given that

$$\Lambda_k^{ST} = \sum_{i=0}^{k} \Lambda_i^S \Lambda_{k-i}^T \quad \text{for } k < n.$$

For notational convenience, we define Λ_i^T, Λ_i^S and Λ_i^{ST} to be zero for $i < 0$.

Note that

$$\begin{aligned}
\Lambda_n^{ST} &= -\frac{1}{n}\left[(S_1+T_1)\Lambda_{n-1}^{ST} + (S_2+T_2)\Lambda_{n-2}^{ST} + \cdots + (S_n+T_n)\right] \\
&= -\frac{1}{n}\left[(S_1+T_1)\sum_{i=0}^{n-1}\Lambda_i^S\Lambda_{n-1-i}^T + (S_2+T_2)\sum_{i=0}^{n-2}\Lambda_i^S\Lambda_{n-2-i}^T + \cdots + (S_n+T_n)\right] \\
&= -\frac{1}{n}\left[S_1\sum_{i=0}^{n-1}\Lambda_i^S\Lambda_{n-1-i}^T + S_2\sum_{i=0}^{n-2}\Lambda_i^S\Lambda_{n-2-i}^T + \cdots + S_n\right. \\
&\quad \left. + T_1\sum_{i=0}^{n-1}\Lambda_i^S\Lambda_{n-1-i}^T + T_2\sum_{i=0}^{n-2}\Lambda_i^S\Lambda_{n-2-i}^T + \cdots + T_n\right] \\
&= -\frac{1}{n}\left[\sum_{i=0}^{n-1}\Lambda_{n-1-i}^T(S_1\Lambda_i^S + S_2\Lambda_{i-1}^S + \cdots + S_{i+1})\right. \\
&\quad \left. + \sum_{i=0}^{n-1}\Lambda_i^S(T_1\Lambda_{n-1-i}^T + T_2\Lambda_{n-2-i}^T + \cdots + T_{n-i})\right] \\
&= -\frac{1}{n}\left[\sum_{i=0}^{n-1}\Lambda_{n-1-i}^T\cdot(-i-1)\Lambda_{i+1}^S + \sum_{i=0}^{n-1}\Lambda_i^S\cdot(-n+i)\Lambda_{n-i}^T\right] \\
&= -\frac{1}{n}\left[(-n)\Lambda_n^S + \sum_{i=0}^{n-2}(-i-1)\Lambda_{i+1}^S\Lambda_{n-1-i}^T + \sum_{i=1}^{n-1}(-n+i)\Lambda_i^S\Lambda_{n-i}^T + (-n)\Lambda_n^T\right] \\
&= -\frac{1}{n}\left[(-n)\Lambda_n^S + \sum_{i=1}^{n-1}((-i)+(-n+i))\Lambda_i^S\Lambda_{n-i}^T + (-n)\Lambda_n^T\right] \\
&= \sum_{i=0}^{n}\Lambda_s^S\Lambda_{n-i}^T.
\end{aligned}$$

This concludes the proof of the theorem. □

Note that an alternative expression for Theorem 1 is

$$\begin{vmatrix}
S_1+T_1 & 1 & 0 & \cdots & 0 & 0 \\
S_2+T_2 & S_1+T_1 & 2 & \cdots & 0 & 0 \\
S_3+T_3 & S_2+T_2 & S_1+T_1 & \cdots & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
S_{k-1}+T_{k-1} & S_{k-2}+T_{k-2} & S_{k-3}+T_{k-3} & \cdots & S_1+T_1 & k-1 \\
S_k+T_k & S_{k-1}+T_{k-1} & S_{k-2}+T_{k-2} & \cdots & S_2+T_2 & S_1+T_1
\end{vmatrix}$$

$$= \sum_{i=0}^{k}\binom{k}{i}\begin{vmatrix}
S_1 & 1 & 0 & \cdots & 0 \\
S_2 & S_1 & 2 & \cdots & 0 \\
S_3 & S_2 & S_1 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
S_{i-1} & S_{i-2} & S_{i-3} & \cdots & i-1 \\
S_i & S_{i-1} & S_{i-2} & \cdots & S_1
\end{vmatrix} \cdot \begin{vmatrix}
T_1 & 1 & 0 & \cdots & 0 \\
T_2 & T_1 & 2 & \cdots & 0 \\
T_3 & T_2 & T_1 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
T_{k-i-1} & T_{k-i-2} & T_{k-i-3} & \cdots & k-i-1 \\
T_{k-i} & T_{k-i-1} & T_{k-i-2} & \cdots & T_1
\end{vmatrix}.$$

Thus, Theorem 1 can be viewed as an extension of the binomial theorem

$$(S+T)^k = \sum_{i=0}^{k}\binom{k}{i}S^iT^{k-i},$$

which follows as a special case when setting $S_1 = S$, $T_1 = T$, and $S_i = T_i = 0$ for $i = 2, 3, \ldots, k$.

Note that Theorem 1 also provides an alternative proof for Proposition 1 (by induction). Our concern in this work is to have one of the sets $\{S_i\}$ or $\{T_i\}$ as power polynomials and the other one as parameters. In such case, Theorem 1 is reduced to the following corollary:

Corollary 1. *Let $x_1, x_2, \ldots, x_n$ be n variables and $s_1, s_2, \ldots, s_k$ be k given parameters. Then,*

$$\mathcal{F}_k(s_1 + \mathcal{S}_1,\ s_2 + \mathcal{S}_2,\ \ldots,\ s_k + \mathcal{S}_k) = \sum_{i=0}^{\min\{n,\,k\}} \mathcal{F}_{k-i}(s_1, s_2, \ldots, s_{k-i}) \cdot \Lambda_i\ . \tag{3.5}$$

4. Solving composite power polynomial equations

In this section, we discuss the solution of the three types of composite power polynomial equations that were introduced in Section 1. We establish the basic method when discussing Type A polynomial systems and then extend the method to Type B and Type C polynomial systems.

4.1. We first consider the following system of composite power polynomial equations:

$$\begin{cases} x_1 + x_2 + \cdots + x_k - x_{k+1} - \cdots - x_n = s_1, \\ x_1^2 + x_2^2 + \cdots + x_k^2 - x_{k+1}^2 - \cdots - x_n^2 = s_2, \\ x_1^3 + x_2^3 + \cdots + x_k^3 - x_{k+1}^3 - \cdots - x_n^3 = s_3, \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\ x_1^n + x_2^n + \cdots + x_k^n - x_{k+1}^n - \cdots - x_n^n = s_n. \end{cases} \tag{4.1}$$

By moving the terms with the negative sign from the left side to the right side, we can rewrite the above equation array as

$$\begin{cases} \sum_{i=1}^{k} x_i = s_1 + \sum_{j=k+1}^{n} x_j, \\ \sum_{i=1}^{k} x_i^2 = s_2 + \sum_{j=k+1}^{n} x_j^2, \\ \sum_{i=1}^{k} x_i^3 = s_3 + \sum_{j=k+1}^{n} x_j^3, \\ \qquad\qquad \vdots \\ \sum_{i=1}^{k} x_i^n = s_n + \sum_{j=k+1}^{n} x_j^n. \end{cases} \tag{4.2}$$

Applying Corollary 1, we conclude that for $k < l \leq n$

$$\begin{aligned} 0 = \Lambda_l(x_1, \ldots, x_k) &= \mathcal{F}_l\left(s_1 + \sum_{j=k+1}^{n} x_j, \ldots, s_l + \sum_{j=k+1}^{n} x_j^l\right) \\ &= \sum_{i=0}^{\min\{l,\,n-k\}} \mathcal{F}_{l-i}(s_1, \ldots, s_{l-i})\Lambda_i(x_{k+1}, \ldots, x_n). \end{aligned}$$

Without loss of generality we can assume that $n - k \leq k$ (when $n - k > k$, we can multiply the equation system (4.1) by -1 and convert it to the case $n - k < k$).

For $l = k+1, k+2, \ldots, n$, this leads to the equation array

$$(4.3)\quad \begin{cases} \sum_{i=1}^{n-k} \mathcal{F}_{k+1-i}(s_1, \cdots, s_{k+1-i})\Lambda_i(x_{k+1}, \ldots, x_n) = -\mathcal{F}_{k+1}(s_1, \ldots, s_{k+1}), \\ \sum_{i=1}^{n-k} \mathcal{F}_{k+2-i}(s_1, \ldots, s_{k+2-i})\Lambda_i(x_{k+1}, \ldots, x_n) = -\mathcal{F}_{k+2}(s_1, \ldots, s_{k+2}), \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\ \sum_{i=1}^{n-k} \mathcal{F}_{n-i}(s_1, \ldots, s_{n-i})\Lambda_i(x_{k+1}, \ldots, x_n) = -\mathcal{F}_n(s_1, \ldots, s_n), \end{cases}$$

where the only unknowns are $\{\Lambda_i(x_{k+1}, x_{k+2}, \ldots, x_n)\}_{i=1}^{n-k}$.

Note that (4.3) is a *linear Toeplitz* problem and can be solved with computational complexity $O((n-k)^2)$ [4]. Proposition 3, which will be shown in the next subsection, implies that there exists *at most one* nontrivial solution $\{x_1, x_2, \ldots, x_n\}$ (in this case, a nontrivial solution is a solution that satisfies $\{x_1, x_2, \ldots, x_k\} \cap \{x_{k+1}, x_{k+2}, \ldots, x_n\} = \emptyset$). When the linear equation system (4.3) is nonsingular, we can solve for $\{\Lambda_i(x_{k+1}, x_{k+2}, \ldots, x_n)\}_{i=1}^{n-k}$, which allows us to obtain $x_{k+1}, x_{k+2}, \ldots, x_n$ as the roots of the characteristic polynomial in (2.6); finally, we can substitute the result into the original system (4.1) and solve for $x_1, x_2, \ldots, x_k$. Note, however, that when the linear system (4.3) is singular, the system (4.1) only has trivial solutions; i.e., there exists $x_i = x_j$, $1 \le i \le k, k < j \le n$. In this case we can reduce both $\{x_1, x_2, \ldots, x_k\}$ and $\{x_{k+1}, x_{k+2}, \ldots, x_n\}$ by one element and repeat the procedure. We are thus ready to present an algorithmic procedure.

Solving procedure for Type A polynomial systems.

(1) Calculate $\mathcal{F}_1(s_1), \mathcal{F}_2(s_1, s_2), \ldots, \mathcal{F}_n(s_1, s_2, \ldots, s_n)$ in light of Newton's identities (2.4).
(2) Solve for $\Lambda_i(x_{k+1}, \ldots, x_n)$, $i = 1, 2, \ldots, n-k$, in light of equation array (4.3).
(3) If (4.3) is singular, then set $k \leftarrow k-1$, $n \leftarrow n-2$ and go to Step 2.
(4) Using $\Lambda_i(x_{k+1}, \ldots, x_n)$, $i = 1, 2, \ldots, n-k$, find the roots of (2.6) to obtain $x_{k+1}, \ldots, x_n$.
(5) Solve for $\mathcal{S}_i(x_{k+1}, \ldots, x_n)$, $i = 1, 2, \ldots, k$, in light of Newton's identities (2.4) (or using the definition in (2.1)).
(6) Set $\mathcal{S}_i(x_1, x_2, \ldots, x_k) \leftarrow s_i + \mathcal{S}_i(x_{k+1}, x_{k+2}, \ldots, x_n)$ for $i = 1, 2, \ldots, k$, and solve for $\Lambda_i(x_1, \ldots, x_k)$, $i = 1, 2, \ldots, k$, in light of Newton's identities (2.4).
(7) Using $\Lambda_i(x_1, \ldots, x_k)$, $i = 1, 2, \ldots, k$, find the roots of (2.6) to obtain $x_1, \ldots, x_k$.

Note that steps (2) and (3) above can also be replaced by the use of the Berlekamp-Massey algorithm with overall complexity $O((n-k)^2)$ (cf. [3]) instead of $O((n-k)^3)$ as above.

4.2. We now consider the more general scenario

$$(4.4)\quad \begin{cases} c_1x_1 + c_2x_2 + \cdots + c_nx_n = s_1, \\ c_1x_1^2 + c_2x_2^2 + \cdots + c_nx_n^2 = s_2, \\ c_1x_1^3 + c_2x_2^3 + \cdots + c_nx_n^3 = s_3, \\ \qquad\qquad\qquad\qquad \vdots \\ c_1x_1^n + c_2x_2^n + \cdots + c_nx_n^n = s_n, \end{cases}$$

where $s_1, s_2, \ldots, s_n$ are given parameters and $c_i \in \{-1, 0, 1\}$ and x_i, $i = 1, 2, \ldots, n$, are the unknowns. We are interested in finding all nontrivial solutions, i.e., in

finding all solutions $\{(c_i, x_i)\}_{i=1}^n$ such that $x_i \neq 0, c_i \neq 0$ and $\{x_i : c_i = 1\} \cap \{x_j : c_j = -1\} = \emptyset$, disregarding the ordering within either set and disregarding the values $\{x_m : c_m = 0\}$.

We first consider the case where k variables $\{x_1, x_2, \ldots, x_k\}$ have coefficient values "1" and l variables $\{x_{k+1}, x_{k+2}, \ldots, x_{k+l}\}$ have coefficient values "-1" ($k + l \leq n$) so that (4.4) (parameterized by (k, l)) reduces to

$$\begin{cases} \sum_{i=1}^{k} x_i - \sum_{j=k+1}^{k+l} x_j &= s_1, \\ \sum_{i=1}^{k} x_i^2 - \sum_{j=k+1}^{k+l} x_j^2 &= s_2, \\ \sum_{i=1}^{k} x_i^3 - \sum_{j=k+1}^{k+l} x_j^3 &= s_3, \\ & \vdots \\ \sum_{i=1}^{k} x_i^n - \sum_{j=k+1}^{k+l} x_j^n &= s_n. \end{cases} \tag{4.5}$$

Proposition 3. *Consider the set of equation systems in* (4.5) *parameterized by* (k, l) *(where* $k + l \leq n$*).*

(i) Among all equation systems (4.5) *satisfying* $k - l = t$ *(for some* $-n \leq t \leq n$*), there exists at most one nontrivial solution (ignoring order).*

(ii) If there exists a nontrivial solution for some (k, l) $(k + l \leq n)$*, then there do not exist alternative nontrivial solutions parameterized within* $\{(k', l') : l' \leq n - k \text{ and } k' \leq n - l\}$.

Proof. (*i*) Suppose that there exist two nontrivial solutions $(x_1, x_2, \ldots, x_{k_1+l_1})$ and $(y_1, y_2, \ldots, y_{k_2+l_2})$ with (k_1, l_1) and (k_2, l_2) so that $k_1 - l_1 = k_2 - l_2$. Then, we have

$$\begin{cases} \sum_{i=1}^{k_1} x_i - \sum_{j=k_1+1}^{k_1+l_1} x_j &= \sum_{i=1}^{k_2} y_i - \sum_{j=k_2+1}^{k_2+l_2} y_j, \\ \sum_{i=1}^{k_1} x_i^2 - \sum_{j=k_1+1}^{k_1+l_1} x_j^2 &= \sum_{i=1}^{k_2} y_i^2 - \sum_{j=k_2+1}^{k_2+l_2} y_j^2, \\ \sum_{i=1}^{k_1} x_i^3 - \sum_{j=k_1+1}^{k_1+l_1} x_j^3 &= \sum_{i=1}^{k_2} y_i^3 - \sum_{j=k_2+1}^{k_2+l_2} y_j^3, \\ & \vdots \\ \sum_{i=1}^{k_1} x_i^n - \sum_{j=k_1+1}^{k_1+l_1} x_j^n &= \sum_{i=1}^{k_2} y_i^n - \sum_{j=k_2+1}^{k_2+l_2} y_j^n. \end{cases} \tag{4.6}$$

We can reorganize the above equation array to

$$\begin{cases} \sum_{i=1}^{k_1} x_i + \sum_{j=k_2+1}^{k_2+l_2} y_j &= \sum_{i=1}^{k_2} y_i + \sum_{j=k_1+1}^{k_1+l_1} x_j, \\ \sum_{i=1}^{k_1} x_i^2 + \sum_{j=k_2+1}^{k_2+l_2} y_j^2 &= \sum_{i=1}^{k_2} y_i^2 + \sum_{j=k_1+1}^{k_1+l_1} x_j^2, \\ \sum_{i=1}^{k_1} x_i^3 + \sum_{j=k_2+1}^{k_2+l_2} y_j^3 &= \sum_{i=1}^{k_2} y_i^3 + \sum_{j=k_1+1}^{k_1+l_1} x_j^3, \\ & \vdots \\ \sum_{i=1}^{k_1} x_i^n + \sum_{j=k_2+1}^{k_2+l_2} y_j^n &= \sum_{i=1}^{k_2} y_i^n + \sum_{j=k_1+1}^{k_1+l_1} x_j^n. \end{cases} \tag{4.7}$$

Note that $k_1 - l_1 = k_2 - l_2$ implies $k_1 + l_2 = k_2 + l_1$, and

$$k_1 + l_2 = \frac{(k_1 + l_2) + (k_2 + l_1)}{2} = \frac{(k_1 + l_1) + (k_2 + l_2)}{2} \leq \frac{n + n}{2} = n\,.$$

Proposition 2 shows that $\{x_1, \ldots, x_{k_1}, y_{k_2+1}, \ldots, y_{k_2+l_2}\}$=$\{y_1, \ldots, y_{k_2}, x_{k_1+1}, \ldots, x_{k_1+l_1}\}$ (ignoring order). In conjunction with the fact that $\{x_1, \ldots, x_{k_1}\}$ cannot overlap $\{x_{k_1+1}, \ldots, x_{k_1+l_1}\}$ and $\{y_1, \ldots, y_{k_2}\}$ cannot overlap $\{y_{k_2+1}, \ldots, y_{k_2+l_2}\}$, this results in $\{x_1, \ldots, x_{k_1}\} = \{y_1, \ldots, y_{k_2}\}$ and $\{x_{k_1+1}, \ldots, x_{k_1+l_1}\} = \{y_{k_2+1}, \ldots, y_{k_2+l_2}\}$.

(*ii*) Assume that there exists an alternative solution parameterized by (k', l'). Then we obtain a similar equation array to (4.7) with $(k_1, l_1) = (k, l)$ and $(k_2, l_2) = (k', l')$. Note that there are $k + l'$ terms on the left side while there are $k' + l$ terms on the right side. Since the case $k + l' = k' + l$ has been shown in (*i*), we just consider $k + l' \neq k' + l$. In such cases, Proposition 2 indicates that the side in (4.7) which contains more terms must be trivial. Thus, it does not have an alternative nontrivial solution. □

Proposition 3 immediately allows us to characterize the number of (nontrivial) solutions in the system of equations in (4.4), as identified by the following proposition.

Proposition 4. *The system of equations* (4.4) *has at most* $n + 1$ *nontrivial solutions.*

Proof. We first show that for a given l there exists at most one nontrivial solution among all choices of k. This is because part (*ii*) of Proposition 3 can be invoked for any (k, l) and (k', l) such that $k + l \leq n$ and $k' + l \leq n$. We thus conclude the proposition since l can only be chosen to be 0, 1, 2, ..., n. □

Assume for now that $k \geq l$ (where k represents the number of "+1" coefficients and l the number of "−1" coefficients). We observe that, for any k such that $k + l \leq n$, we can choose the following *common* equation array in solving for $\{\Lambda_i(x_{k+1}, x_{k+2}, \ldots, x_{k+l})\}_{i=1}^{l}$:

(4.8)
$$\begin{cases} \sum_{i=1}^{l} \mathcal{F}_{n-l+1-i}(s_1, \ldots, s_{n-l+1-i})\Lambda_i(x_{k+1}, \ldots, x_{k+l}) & = -\mathcal{F}_{n-l+1}(s_1, \ldots, s_{n-l+1}), \\ \sum_{i=1}^{l} \mathcal{F}_{n-l+2-i}(s_1, \ldots, s_{n-l+2-i})\Lambda_i(x_{k+1}, \ldots, x_{k+l}) & = -\mathcal{F}_{n-l+2}(s_1, \ldots, s_{n-l+2}), \\ & \vdots \\ \sum_{i=1}^{l} \mathcal{F}_{n-i}(s_1, \ldots, s_{n-i})\Lambda_i(x_{k+1}, \ldots, x_{k+l}) & = -\mathcal{F}_{n}(s_1, \ldots, s_{n}). \end{cases}$$

If a solution for $\{\Lambda_i(x_{k+1}, x_{k+2}, \ldots, x_{k+l})\}_{i=1}^{l}$ exists, we can obtain a proper solution $x_{k+1}, x_{k+2}, \ldots, x_{k+l}$ with respect to the given l. Note that if $\Lambda_l(x_{k+1}, x_{k+2}, \ldots, x_{k+l}) = 0$ (which implies that one of the roots is zero), then the resulting solution must be trivial and thus we need not go any further. Once we have $x_{k+1}, x_{k+2}, \ldots, x_{k+l}$, we can compute $\tilde{s}_i \stackrel{\triangle}{=} s_i + \sum_{j=k+1}^{k+l} x_j^i$, $i = 1, 2, \ldots, n-l$, and then solve the following equation array for the remaining nontrivial unknowns $x_1, x_2, \ldots, x_k$ with $(k \leq n - l)$:

(4.9)
$$\begin{cases} \sum_{i=1}^{k} x_i & = \tilde{s}_1, \\ \sum_{i=1}^{k} x_i^2 & = \tilde{s}_2, \\ & \vdots \\ \sum_{i=1}^{k} x_i^{n-l} & = \tilde{s}_{n-l}\ . \end{cases}$$

Proposition 2 indicates that $\{y_i\}_{i=1}^{n-l} = \{x_i\}_{i=1}^{k} \cup \{0\}$, where $\{y_i\}_{i=1}^{n-l}$ are the unique (up to permutation) solutions of

(4.10)
$$\begin{cases} \sum_{i=1}^{n-l} y_i & = \tilde{s}_1, \\ \sum_{i=1}^{n-l} y_i^2 & = \tilde{s}_2, \\ & \vdots \\ \sum_{i=1}^{n-l} y_i^{n-l} & = \tilde{s}_{n-l}. \end{cases}$$

Therefore, we only need to solve the system of equations (4.10) and keep the nonzero roots of the characteristic polynomial in (2.6) as the solution of $x_1, x_2, \ldots, x_k$.

Clearly, when $k < l$, we can simply multiply both sides of the equation array (4.5) by -1 and apply the procedure again. We now summarize our algorithmic procedure.

Solving procedure for Type B polynomial systems.

(1) Calculate $\mathcal{F}_1(s_1), \mathcal{F}_2(s_1, s_2), \ldots, \mathcal{F}_n(s_1, s_2, \ldots, s_n)$ using Newton's identities (2.4).
(2) For $l = 0, 1, 2, \ldots, \lfloor n/2 \rfloor$, **Call Routine**. End
(3) Multiply the system of equations by -1, i.e., set $s_1 \leftarrow -s_1$, $s_2 \leftarrow -s_2$, $\ldots$, $s_n \leftarrow -s_n$.
(4) Calculate $\mathcal{F}_1(s_1), \mathcal{F}_2(s_1, s_2), \ldots, \mathcal{F}_n(s_1, s_2, \ldots, s_n)$ using Newton's identities (2.4).
(5) For $l = 0, 1, 2, \ldots, \lfloor (n-1)/2 \rfloor$, **Call Routine**. End

Routine.

(1) Solve equation array (4.8). If singular or $\Lambda_l(x_{n-l+1}, x_{n-l+2}, \ldots, x_n) = 0$, continue with the next loop.
(2) Solve for roots of (2.6) using the obtained $\{\Lambda_i(x_{n-l+1}, x_{n-l+2}, \ldots, x_n)\}_{i=1}^{l}$. If no solution exists in the given field, continue with the next loop.
(3) Calculate $\{\tilde{s}_i \stackrel{\triangle}{=} s_i + \mathcal{S}_i(x_{n-l+1},\ x_{n-l+2},\ \ldots,\ x_n)\}_{i=1}^{n-l}$ using the obtained $\{x_{n-l+1},\ x_{n-l+2},\ \ldots,\ x_n\}$.
(4) Calculate $\{\mathcal{F}_i(\tilde{s}_1,\ \tilde{s}_2,\ \ldots,\ \tilde{s}_i)\}_{i=1}^{n-l}$ in light of Newton's identities (2.4).
(5) Solve for the roots of (2.6) using the obtained $\{\Lambda_i(x_1,\ x_2,\ \ldots,\ x_{n-l})\}_{i=1}^{n-l}$. Throw away all zero roots.

4.3. In this subsection, we show that the binomial decomposition rule is also applicable to Type C polynomial systems, given by

$$\left\{\begin{array}{lllllll} x_1 & +x_2 & +\cdots & +x_{n-1} & -\alpha x_n & = s_1, \\ x_1^2 & +x_2^2 & +\cdots & +x_{n-1}^2 & -\alpha x_n^2 & = s_2, \\ x_1^3 & +x_2^3 & +\cdots & +x_{n-1}^3 & -\alpha x_n^3 & = s_3, \\ & & & & & \vdots \\ x_1^n & +x_2^n & +\cdots & +x_{n-1}^n & -\alpha x_n^n & = s_n, \end{array}\right. \tag{4.11}$$

where α is an arbitrary value other than $0, 1, -1$ (with $\alpha \in \{0, 1, -1\}$ the above equation system reduces to trivial cases of Type A polynomial systems).

Since $\Lambda_n(x_1, x_2, \ldots, x_{n-1}) = 0$, we have

$$\begin{aligned} 0 &= \mathcal{F}_n(s_1 + \alpha x_n, s_2 + \alpha x_n^2, \ldots, s_n + \alpha x_n^n) \\ &= \sum_{i=0}^{n} \mathcal{F}_i(s_1, \ldots, s_i) \mathcal{F}_{n-i}(\alpha x_n, \ldots, \alpha x_n^{n-i}) \,. \end{aligned} \tag{4.12}$$

We can further simplify the above equality in light of the following proposition.

Proposition 5.

$$\mathcal{F}_k(\alpha x, \alpha x^2, \ldots, \alpha x^k) = \binom{\alpha}{k}(-x)^k, \tag{4.13}$$

where $\binom{\alpha}{k} \triangleq \frac{\prod_{i=0}^{k-1}(\alpha - i)}{k!}$ *is defined for an arbitrary value* α *and an integer* k *(for* $k = 0$ *we set* $\binom{\alpha}{0} = 1$*).*

Proof. The proof is by induction. Trivially, the equality holds when $k = 1$. Assuming that the equality holds for all $k < K$, we need to show that it holds for $k = K$. As indicated by Newton's identities, we have

$$\begin{aligned}
\mathcal{F}_K(\alpha x, \alpha x^2, \ldots, \alpha x^K) &= -\frac{1}{K}\sum_{i=0}^{K-1} \alpha x^{K-i}\mathcal{F}_i(\alpha x, \alpha x^2, \ldots, \alpha x^i) \\
&= -\frac{\alpha}{K}\sum_{i=0}^{K-1} x^{K-i}\binom{\alpha}{i}(-x)^i \\
&= -x^K\frac{\alpha}{K}\sum_{i=0}^{K-1}(-1)^i\binom{\alpha}{i} \\
&= -x^K\frac{\alpha}{K}(-1)^{K-1}\binom{\alpha-1}{K-1} \\
&= \binom{\alpha}{K}(-x)^K,
\end{aligned}$$

where the equality

$$\sum_{i=0}^{k}(-1)^i\binom{\alpha}{i} = (-1)^k\binom{\alpha-1}{k}$$

can be easily shown, again by induction. We thus conclude the proposition. □

Thus, (4.12) can be further reduced to the form

$$0 = \sum_{i=0}^{n}\binom{\alpha}{i}\mathcal{F}_{n-i}(s_1, \ldots, s_{n-i})(-x_n)^i. \tag{4.14}$$

Since there exist up to n distinct roots of x_n in (4.14), the equation system (4.11) has at most n distinct solution sets (ignoring the order of $x_1, x_2, \ldots, x_{n-1}$). We summarize the discussion in this subsection with the following algorithmic procedure.

Solving procedure for Type C polynomial systems.

(1) Compute $\{\binom{\alpha}{i}\}_{i=1}^{n-1}$ recursively and $\{\mathcal{F}_i(\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_i)\}_{i=1}^{n-1}$ in light of Newton's identities in (2.4).
(2) Solve for x_n using (4.14).
(3) For each distinct root of x_n, do:
   - Compute $\{\alpha x_n^i\}_{i=1}^{n-1}$ recursively.
   - Compute $\{\Lambda_i(x_1, x_2, \ldots, x_{n-1})\}_{i=1}^{n-1}$ in light of Newton's identities (2.4).
   - Solve for the roots of (2.6) to obtain $x_1, x_2, \ldots, x_{n-1}$.

5. Examples

We consider the following example in GF(31):

$$\begin{cases} c_1x_1+ & c_2x_2+ & c_3x_3+ & c_4x_4+ & c_5x_5 & = 1, \\ c_1x_1^2+ & c_2x_2^2+ & c_3x_3^2+ & c_4x_4^2+ & c_5x_5^2 & = 1, \\ c_1x_1^3+ & c_2x_2^3+ & c_3x_3^3+ & c_4x_4^3+ & c_5x_5^3 & = 4, \\ c_1x_1^4+ & c_2x_2^4+ & c_3x_3^4+ & c_4x_4^4+ & c_5x_5^4 & = 1, \\ c_1x_1^5+ & c_2x_2^5+ & c_3x_3^5+ & c_4x_4^5+ & c_5x_5^5 & = 1, \end{cases}$$

where $c_i \in \{0, \pm 1\}$ and $x_i \in$GF(31), $i = 1, 2, 3, 4, 5$.

We proceed to demonstrate the procedure of identifying nontrivial solutions to the above Type B polynomial system.

(1) Using (2.4), we obtain $\mathcal{F}_1 = 30$, $\mathcal{F}_2 = 0$, $\mathcal{F}_3 = 30$, $\mathcal{F}_4 = 1$, $\mathcal{F}_5 = 0$.

(2) Case $l = 0$: x_1, x_2, x_3, x_4, x_5 are the roots of the single variable polynomial $x^5 + 30x^4 + 30x^2 + x = 0$, which can be factorized as

$$x^5 + 30x^4 + 30x^2 + x = x(x-1)^2(x+6)(x-5) \, .$$

Therefore, we have the following nontrivial solution:

$$\{(1, 1), \ (1, 1), \ (1, 25), \ (1, 5)\} \, .$$

(3) Case $l = 1$: We have from (4.8)

$$\mathcal{F}_4\Lambda_1(x_5) = -\mathcal{F}_5 \ \ \Rightarrow \ \ \Lambda_1(x_5) = -x_5 = 0 \, .$$

Thus, we conclude that this case yields no nontrivial solution.

(4) Case $l = 2$: We have from (4.8)

$$\begin{cases} 30\Lambda_1(x_4, x_5) = -1, \\ 30\Lambda_2(x_4, x_5) + \Lambda_1(x_4, x_5) = 0 \end{cases} \Rightarrow \begin{cases} \Lambda_1(x_4, x_5) = 1, \\ \Lambda_2(x_4, x_5) = 1. \end{cases}$$

Hence, x_4, x_5 are the roots of $x^2 + x + 1 = 0$. Solving for the roots of this polynomial, we obtain $\{x_4, x_5\} = \{5, 25\}$. Based on these values for x_4, x_5 and the given system of composite polynomial equations, the power sums of x_1, x_2, x_3 can be easily obtained as

$$\mathcal{S}_1 = 1 + 5 + 25 = 0, \ \ \mathcal{S}_2 = 1 + 5^2 + 25^2 = 0, \ \ \mathcal{S}_3 = 4 + 5^3 + 25^3 = 6 \, .$$

From these we can obtain $\{\Lambda_i(x_1, x_2, x_3)\}_{i=1}^3$ as

$$\Lambda_1 = 0, \ \ \Lambda_2 = 0, \ \ \Lambda_3 = -2 \, .$$

Thus, $\{x_1, x_2, x_3\}$ are the roots of the polynomial $x^3 - 2 = 0$. Solving, we obtain $\{x_1, x_2, x_3\} = \{4, 7, 20\}$. Therefore, the second nontrivial solution is given by

$$\{(1, 4), \ (1, 7), \ (1, 20), \ (-1, 5), \ (-1, 25)\}.$$

(5) We multiply each of s_1, s_2, s_3, s_4, s_5 by -1. Consequently, we obtain

$$\mathcal{F}_1 = 1, \;\; \mathcal{F}_2 = 1, \;\; \mathcal{F}_3 = 2, \;\; \mathcal{F}_4 = 2, \;\; \mathcal{F}_5 = 2 \; .$$

(6) Case $k = 0$: x_1, x_2, x_3, x_4, x_5 are the roots of the polynomial $x^5 + x^4 + x^3 + 2x^2 + 2x + 2 = 0$, which can be calculated to be 5, 11, 24, 25, 27. Thus, we obtain the third nontrivial solution

$$\{(-1, 5), \;\; (-1, 11), \;\; (-1, 24), \;\; (-1, 25), \;\; (-1, 27)\} \; .$$

(7) Case $k = 1$: We have from (4.8)

$$\mathcal{F}_4 \Lambda_1(x_5) = -\mathcal{F}_5 \;\; \Rightarrow \;\; \Lambda_1(x_5) = -x_5 = -2/2 \;\; \Rightarrow \;\; x_5 = 1 \; .$$

Based on the above value of x_5 and the given system of composite polynomial equations, we can easily compute the values of the power sums with respect to x_1, x_2, x_3, x_4 as

$$\mathcal{S}_1 = -1 + 1 = 0, \;\; \mathcal{S}_2 = -1 + 1 = 0, \;\; \mathcal{S}_3 = -4 + 1 = 28, \;\; \mathcal{S}_4 = -1 + 1 = 0 \; .$$

We next compute the values of the symmetric polynomials as $\Lambda_1 = 0, \;\; \Lambda_2 = 0, \;\; \Lambda_3 = 1, \;\; \Lambda_4 = 0$. Thus, x_1, x_2, x_3, x_4 must be the roots of equation $x^4 + x = 0$, which are given by 0, 6, 26, 30. Therefore, the forth nontrivial solution is

$$\{(-1, 6), \;\; (-1, 26), \;\; (-1, 30), \;\; (1, 1)\} \; .$$

(8) Case $k = 2$: We have from (4.8)

$$\begin{cases} \Lambda_2(x_4, x_5) + 2\Lambda_1(x_4, x_5) = -2, \\ 2\Lambda_2(x_4, x_5) + 2\Lambda_1(x_4, x_5) = -2 \end{cases} \quad \Rightarrow \quad \begin{cases} \Lambda_1(x_4, x_5) = 30, \\ \Lambda_2(x_4, x_5) = 0 \; . \end{cases}$$

Since $\Lambda_2 = 0$, there does not exist a nontrivial solution for this case.

Alternatively, we consider classical decoding of the $(31, 26)$ Reed-Solomon code with error syndrome [1 1 4 1 1], which is the same as in the above example. The Berlekamp-Massey algorithm is capable of correcting two arbitrary errors; i.e., it is capable of solving the system

$$\begin{cases} c_1 x_1 + c_2 x_2 = 1, \\ c_1 x_1^2 + c_2 x_2^2 = 1, \\ c_1 x_1^3 + c_2 x_2^3 = 4, \\ c_1 x_1^4 + c_2 x_2^4 = 1. \end{cases}$$

The above system does not have a valid solution in GF(31). However, our method is able to generate four candidate code words. In fact, this example illustrates that, under the assumption that errors of the form ± 1 are more likely than arbitrary errors, our method has greater correcting capabilities than the Berlekamp-Massey algorithm.

Acknowledgments

The authors would like to thank Professor Bruce Reznick of the Department of Mathematics at the University of Illinois at Urbana-Champaign for helpful discussions and for making many valuable suggestions.

References

1. D. Dobbs and R. Hanks, *A Modern Course on the Theory of Equations*, 2nd edition, Polygonal Publishing House, Washington, NJ, 1992. MR 93b:12001
2. Y. Wu and C. N. Hadjicostis, "Non-concurrent fault detection and identification using encoded Petri net models of discrete event systems," *Proceedings of the 2002 IEEE Conf. on Decision and Control*, vol. 4, pp. 4018–4023, Las Vegas, Nevada, 2002.
3. R. E. Blahut, *Algebraic Codes for Data Transmission*, Cambridge University Press, Cambridge, UK, 2002.
4. D. Bini and V. Y. Pan, *Polynomial and Matrix Computations, vol. 1: Fundamental Algorithms*, Birkhäuser Boston, Cambridge, MA, 1994. MR 95k:65003

Coordinated Science Laboratory and Department of Electrical and Computer Engineering, University of Illinois, Urbana-Champaign, Illinois 61801

Current address: 139 Coordinated Science Laboratory, 1308 West Main Street, Coordinated Science Laboratory and Department of Electrical and Computer Engineering, University of Illinois, Urbana-Champaign, Illinois 61801-2307

E-mail address: ywu4@uiuc.edu

Coordinated Science Laboratory and Department of Electrical and Computer Engineering, University of Illinois, Urbana-Champaign, Illinois 61801

Current address: 357 Coordinated Science Laboratory, 1308 West Main Street, Coordinated Science Laboratory and Department of Electrical and Computer Engineering, University of Illinois, Urbana-Champaign, Illinois 61801-2307

E-mail address: chadjic@uiuc.edu