Fast Multi-Sequence Shift-Register Synthesis with the Euclidean Algorithm

Alexander Zeh, Antonia Wachter

Advances in Mathematics of Communications
Volume 5, No. 4, 2011, 667–680
doi:10.3934/amc.2011.5.667

FAST MULTI–SEQUENCE SHIFT–REGISTER SYNTHESIS WITH THE EUCLIDEAN ALGORITHM

Alexander Zeh

Institute of Communications Engineering
Ulm University, Albert-Einstein-Allee 43, 89083 Ulm, Germany
and
Research Center INRIA Saclay - Île-de-France
École Polytechnique, 91128 Palaiseau Cedex, France

Antonia Wachter

Institute of Communications Engineering
Ulm University, Albert-Einstein-Allee 43, 89083 Ulm, Germany
and
Institut de Recherche Mathémathique de Rennes (IRMAR)
Université de Rennes 1, 35042 Rennes Cedex, France

(Communicated by Michael O'Sullivan)

Abstract. Feng and Tzeng's generalization of the Extended Euclidean Algorithm synthesizes the shortest–length linear feedback shift–register for $s \geq 1$ sequences, where each sequence has the same length n. In this contribution, it is shown that Feng and Tzeng's algorithm which solves this multi–sequence shift–register problem has time complexity $\mathcal{O}(sn^2)$. An acceleration based on the Divide and Conquer strategy is proposed and it is proven that subquadratic time complexity is achieved.

1. Introduction

Multi–sequence *linear feedback shift–register* (LFSR) synthesis plays an important role in cryptography and coding theory, e.g. for decoding *Interleaved Reed–Solomon* (IRS) codes [3, 8, 9, 12]. A codeword of an IRS code can be seen as s parallel codewords from Reed–Solomon codes of same length. Transmitting an IRS codes provides s syndrome sequences, which can be used to determine one common error locator polynomial for the s parallel codewords. This error locator polynomial can be interpreted as the connection polynomial of a shift register, which generates *each* of these s syndrome sequences. Another application is decoding of binary cyclic codes up to the Hartmann–Tzeng [7] bound, where multiple sets of consecutive roots result in multiple syndrome sequences of equal length.

Solving the multi–sequence LFSR synthesis problem for $s \geq 1$ sequences means finding the shortest–length LFSR that generates each of the s sequences.

2000 *Mathematics Subject Classification:* Primary: 94A55, 94B15; Secondary: 94B35.

Key words and phrases: Divide and conquer, (extended) Euclidean algorithm, interleaved reed–Solomon codes, fast algorithms, (multi–sequence) shift–register synthesis.

The work of A. Zeh is supported by the German Research Council Deutsche Forschungsgemeinschaft (DFG) under Grant No. Bo867/22-1. The work of A. Wachter is supported by the German Research Council Deutsche Forschungsgemeinschaft (DFG) under Grant No. Bo867/21-1.

Mainly, there are two algorithms for finding the shortest–length LFSR for sequences of equal length n. Both were introduced by Feng and Tzeng: one based on the Berlekamp–Massey Algorithm [5] and one based on the Euclidean Algorithm [4]. A generalization of Feng and Tzeng's Euclidean Algorithm to Euclidean modules was considered in [14].

However, the complexity of [4] has not been analyzed so far. Many efficient algorithms are based on the so–called *Divide and Conquer* (DC) strategy. Assume, a problem of size M is given, then the DC strategy splits the problem into two halves, each of size $M/2$. The structure of these halves should be the same as the original problem. The calculation can be accelerated if [1, 2, 6]

1. there are algorithms with less than half of the complexity for the divided problems,
2. and they can be combined into the solution of the whole problem with low complexity.

In this contribution, we accelerate Feng and Tzeng's *(Extended) Generalized Euclidean Algorithm* ((E)GEA) [4] using the DC strategy. We show that the EGEA has complexity $\mathcal{O}(sn^2)$ when solving the multi–sequence shift–register problem. Our fast algorithm has subquadratic complexity $\mathcal{O}(s^2 n \log^2 sn)$.

This paper is organized as follows. In Section 2, we state the problem and explain Feng and Tzeng's original EGEA [4]. Section 3 provides and proves our fast algorithm for solving the multi–sequence LFSR synthesis problem for sequences of equal length. In Section 4, we analyze the complexity of both, the original and the fast algorithm and come to a conclusion in Section 5.

2. The (extended) generalized Euclidean algorithm

2.1. Problem statement.

Let $\mathbb{F}_q$ denote a finite field of order q and $\mathbb{F}_q[x]$ stands for the set of all univariate polynomials in the indeterminate x over $\mathbb{F}_q$.

Problem 2.1 (Multi–sequence shift–register synthesis of equal length)**.** *Given $s \geq 1$ sequences $\mathbf{S}^{(i)} = (S_0^{(i)}, S_1^{(i)}, \ldots, S_{n-1}^{(i)})$, $i = 0, \ldots, s-1$, where each $S_j^{(i)} \in \mathbb{F}_q$ of the same length n. Find the connection polynomial $\sigma(x) = \sigma_0 + \sigma_1 x + \cdots + \sigma_{\ell-1} x^{\ell-1} + x^\ell \in \mathbb{F}_q[x]$ of minimal degree ℓ such that:*

$$S_j^{(i)} + \sigma_{\ell-1} \cdot S_{j-1}^{(i)} + \cdots + \sigma_0 \cdot S_{j-\ell}^{(i)} = 0 \tag{1}$$

for all $j = \ell, \ell+1, \ldots, n-1$ and for all $i = 0, \ldots, s-1$.

Similar to the well–known key equation for the single–sequence shift–register synthesis problem, Feng and Tzeng reformulated an equation for Problem 2.1 [4, Equation (3)]. An alternative derivation of this concatenated key equation can be found in [15]. Here, we give the basic idea. Let the assumptions of Problem 2.1 be fulfilled, define a polynomial $S(x) \in \mathbb{F}_q[x]$ with $\deg S(x) < sn$ by

$$S(x) \stackrel{\text{def}}{=} \sum_{i=0}^{s-1} x^{s-1-i} \sum_{j=0}^{n-1} S_j^{(i)} x^{s(n-1-j)}. \tag{2}$$

In [4], it was shown that solving Problem 2.1 is equivalent to the following problem.

Problem 2.2 (Solving Concatenated Key Equation)**.** *Given $s \geq 1$ sequences $\mathbf{S}^{(i)} = (S_0^{(i)}, S_1^{(i)}, \ldots, S_{n-1}^{(i)})$, $i = 0, \ldots, s-1$ of the same length n and hence also the*

polynomial $S(x)$ *with* $\deg S(x) < sn$ *as defined in* (2). *Find a polynomial* $\sigma(x^s) \in \mathbb{F}_q[x]$ *such that*

$$S(x) \cdot \sigma(x^s) \equiv r(x) \mod x^{sn}, \tag{3}$$

where $\deg r(x) < \deg \sigma(x^s)$.

Note that for $s = 1$, Equation (3) is the classical key equation for single–sequence shift–register synthesis.

2.2. OVERVIEW OF THE ALGORITHMS. Figure 1 illustrates the connection between Feng and Tzeng's (E)GEA [4] and the classical *(Extended) Euclidean Algorithm* ((E)EA). The classical single–sequence shift–register synthesis problem (i.e., only *one* sequence is given) can be solved by the (E)EA [13]. Feng and Tzeng's generalization to the (E)GEA solves the multi–sequence shift–register synthesis problem (Problem 2.2) for $s \geq 1$ sequences. In general, the EA is used to cal-

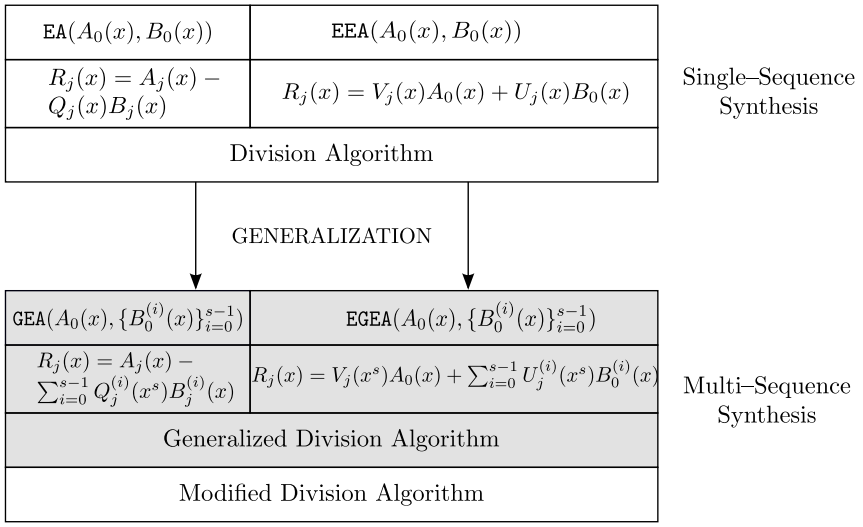

FIGURE 1. Overview of Feng and Tzeng's algorithms. The two parts distinguish between the classical single–sequence problem, which can be solved by the (E)EA, and the multi–sequence problem, which can be solved by the (E)GEA. The basic algorithm of the (E)EA is the usual division of two polynomials. For the (E)GEA, the modified and the generalized division are the equivalent of the division. The EA calculates the GCD of two polynomials and the EEA additionally puts out coefficients to obtain a linear combination of the input polynomials in each step. The EGEA is an extension of the GEA to obtain such coefficients for each of the $s + 1$ input polynomials.

culate the *Greatest Common Divisor* (GCD) of two polynomials $A_0(x), B_0(x)$. The EEA additionally calculates polynomials $V_j(x)$ and $U_j(x)$ to obtain a linear combination of the two input polynomials $A_0(x), B_0(x)$ such that for the remainder $R_j(x) = V_j(x) \cdot A_0(x) + U_j(x) \cdot B_0(x)$ holds in each step j of the algorithm. The basic algorithm of the (E)EA is the usual division algorithm of two polynomials.

The GEA can be seen as the generalization of the EA to $s \geq 1$ sequences. The EGEA extends the GEA by factors in the same way as the EEA extends the EA.

The EGEA solves Problem 2.2 and returns corresponding linear factors $V_j(x^s)$, $\{U_j^{(i)}(x^s)\}_{i=0}^{s-1}$ as the EEA in the single–sequence case. Thereby, the basic algorithms of the EGEA are the so–called modified and generalized division. Note that Feng and Tzeng called the EGEA *Alternative Version of the Generalized Euclidean Algorithm* [4, Section II-D]. For $s = 1$, the EGEA is the EEA.

For a description of the (E)GEA, we give some definitions in the following and explain the subalgorithms shown in Figure 1. We focus on the most important parts and rewrite Feng and Tzeng's algorithms in a compact form. Additional properties (e.g. degree constraints) can be found in [4].

Definition 2.1 (Congruence Class [4])**.** Two polynomials $A(x)$, $B(x) \in \mathbb{F}_q[x]$ are equivalent if and only if

$$\deg A(x) \equiv \deg B(x) \mod s.$$

We denote it by $A(x) \sim B(x)$, otherwise $A(x) \nsim B(x)$. Then, the congruence class $\mathcal{A}$ of degree ν represented by $A(x)$ with $\deg A(x) \mod s = \nu$, is the following set of polynomials:

$$\mathcal{A} = \left\{ f(x) \in \mathbb{F}_q[x] \;\middle|\; \deg f(x) \mod s \equiv \nu \right\}.$$

Definition 2.2 (Set of Representatives)**.** For an integer s, a set of representatives $\mathcal{B}$ is the following set of all s polynomials $\{f^{(i)}(x)\}_{i=0}^{s-1}$:

$$\mathcal{B} = \left\{ \{f^{(i)}(x)\}_{i=0}^{s-1} \in \mathbb{F}_q[x] \;\middle|\; f^{(i)}(x) \sim x^i,\ \forall i \right\}. \tag{4}$$

The modified division algorithm is the basic subalgorithm of the GEA and is given in Algorithm 1.

Theorem 2.3 (Modified Division [4])**.** *Given two polynomials $A(x) \sim B(x)$ in $\mathbb{F}_q[x]$ where $\deg A(x) \geq \deg B(x)$, the modified division (Algorithm 1, `ModDA`) calculates unique polynomials $Q(x^s)$ and $R(x)$ such that*

$$A(x) = Q(x^s) \cdot B(x) + R(x), \tag{5}$$

where $\deg R(x) < \deg B(x)$ if $R(x) \sim B(x)$ and $\deg R(x) \geq \deg B(x)$ might be possible if $R(x) \nsim B(x)$. We call $Q(x^s)$ the quotient and $R(x)$ the remainder.

In Line 4 of Algorithm 1, $\mathrm{lc}(R(x)), \mathrm{lc}(B(x))$ denotes the leading coefficients of $R(x)$ and $B(x)$. Since $R(x) \sim B(x)$, the result of this line is a monomial $\widetilde{Q}(x^s)$ with the indeterminate x^s.

The modified division of two polynomials consists of the first steps of a usual division of the same polynomials. The iterations of the modified division might stop earlier, but never later than the iterations of the usual division. For $s = 1$, the modified division is the same as the usual division of two polynomials.

Algorithm 1: Modified Division Algorithm (`ModDA`)

Input: $A(x)$, $B(x)$, s with $A(x) \sim B(x)$ and $\deg A(x) \geq B(x)$
Initialize: $R(x) \leftarrow A(x)$, $Q(x^s) \leftarrow 0$

$\nu \leftarrow \deg A(x) \mod s$
$\widetilde{\nu} \leftarrow \nu$
while $\widetilde{\nu} = \nu$ ***and*** $\deg R(x) \geq \deg B(x)$ **do**
  $\widetilde{Q}(x^s) \leftarrow \mathrm{lc}(R(x))/\mathrm{lc}(B(x)) \cdot x^{\deg R(x) - \deg B(x)}$ `// leading coefficients`
  $Q(x^s) \leftarrow Q(x^s) + \widetilde{Q}(x^s)$
  $R(x) \leftarrow R(x) - \widetilde{Q}(x^s) \cdot B(x)$
  $\widetilde{\nu} \leftarrow \deg R(x) \mod s$ `// calculate new congruence class`

Output: $R(x)$, $Q(x^s)$

In the following, we give an example of this modified division. This example is the same as the first step of [4, Section II, Example 1], but with more intermediate steps as explanation.

Example 2.1 (Modified Division)**.** The example considers sequences over $\mathbb{F}_2$. Let the input be: $A(x) = x^{11} + x^8 + x^2 + 1$, $B(x) = x^5 + x + 1$ and let $s = 3$. We initialize $R(x) \leftarrow A(x)$, $Q(x^s) \leftarrow 0$ and $\nu \leftarrow 2$. The steps of the algorithm are as follows.

1. $\nu = \widetilde{\nu} = 2$ and $\deg R(x) \geq \deg B(x)$, hence we calculate
   $\widetilde{Q}(x^s) \leftarrow x^6$,
   $Q(x^s) \leftarrow 0 + x^6 = x^6$,
   $R(x) \leftarrow R(x) - (x^{11} + x^7 + x^6) = x^8 + x^7 + x^6 + x^2 + 1$,
   $\widetilde{\nu} \leftarrow 2$.
2. $\nu = \widetilde{\nu} = 2$ and $\deg R(x) \geq \deg B(x)$, hence we calculate
   $\widetilde{Q}(x^s) \leftarrow x^3$,
   $Q(x^s) \leftarrow x^6 + x^3$,
   $R(x) \leftarrow R(x) - (x^8 + x^4 + x^3) = x^7 + x^6 + x^4 + x^3 + x^2 + 1$,
   $\widetilde{\nu} \leftarrow 1$.
3. $\nu = 2 \neq \widetilde{\nu} = 1$ and no further iteration is done.

The output is $R(x) = x^7 + x^6 + x^4 + x^3 + x^2 + 1$ and $Q(x^s) = x^6 + x^3$.

The so–called generalized division applies the modified division repeatedly. The generalized division (together with the modified division) can be seen as the equivalent in the EGEA to the usual division algorithm in the EEA (see Figure 1).

Theorem 2.4 (Generalized Division [4])**.** *Given $s+1$ polynomials $A(x)$, $\overline{\mathcal{B}} = \{B^{(i)}(x)\}_{i=0}^{s-1}$ in $\mathbb{F}_q[x]$, where $\overline{\mathcal{B}}$ is an element of the set of representatives (4) and $\deg A(x) \geq B^{(i)}(x)$ $\forall i = 0, \ldots, s-1$. The generalized division (Algorithm 2, `GenDA`) calculates $s+1$ polynomials $R(x)$, $\{Q^{(i)}(x^s)\}_{i=0}^{s-1}$ such that*

$$A(x) = \sum_{i=0}^{s-1} Q^{(i)}(x^s) \cdot B^{(i)}(x) + R(x), \tag{6}$$

where $\deg R(x) < \deg B^{(\nu)}(x)$, where ν is chosen such that $R(x) \sim B^{(\nu)}(x)$.

Algorithm 2: Generalized Division Algorithm (`GenDA`)

Input: $A(x)$, $\{B_0^{(i)}(x)\}_{i=0}^{s-1}$ with $\deg A(x) \geq \deg B^{(i)}(x)\ \forall i = 0, \ldots, s-1$
Initialize: $R(x) \leftarrow A(x)$, $\{Q^{(i)}(x^s)\}_{i=0}^{s-1} \leftarrow \{0\}_{i=0}^{s-1}$

1 Calculate ν such that $R(x) \sim B^{(\nu)}(x)$
2 **while** $\deg R(x) \geq \deg B^{(\nu)}(x)$ **do**
3 $\quad R(x), \widetilde{Q}(x^s) \leftarrow$ `ModDA` $\left(R(x), B^{(\nu)}(x), s\right)$ `// modified division`
4 $\quad Q^{(\nu)}(x^s) \leftarrow Q^{(\nu)}(x^s) + \widetilde{Q}(x^s)$
5 $\quad$ Calculate ν such that $R(x) \sim B^{(\nu)}(x)$ `// calculate new congr. class`

Output: $R(x)$, $\{Q^{(i)}(x^s)\}_{i=0}^{s-1}$

For an example of the generalized division see [4, Section II, Example 1].

Based on the previous definitions and algorithms, we now describe the GEA and extend it to the EGEA, which synthesizes the shortest–length multi–sequence LFSR. The GEA is defined as follows [4, Equation (8)].

Theorem 2.5 (Generalized Euclidean Algorithm, GEA [4]). *Given $s+1$ polynomials $A_0(x)$, $\overline{\mathcal{B}}_0 = \{B_0^{(i)}(x)\}_{i=0}^{s-1}$ in $\mathbb{F}_q[x]$, where $\overline{\mathcal{B}}_0$ is an element of the set of representatives (4), $A_0(x) \sim B_0^{(\nu_0)}(x)$ and $\deg A_0(x) \geq \deg B_0^{(\nu_0)}(x)$. The GEA repeatedly applies the generalized division (Algorithm 2) to obtain $s+1$ polynomials $R_j(x)$, $\{Q_j^{(i)}(x^s)\}_{i=0}^{s-1}$ such that*

$$R_j(x) = A_j(x) - \sum_{i=0}^{s-1} Q_j^{(i)}(x^s) \cdot B_j^{(i)}(x), \tag{7}$$

for $j = 0, 1 \ldots$, where ν_j is chosen such that $R_j(x) \sim B_j^{(\nu_j)}(x)$ and $\deg R_j(x) < \deg B_j^{(\nu_j)}(x)$. To calculate (7) for $j \geq 1$, let

$$A_j(x) \leftarrow B_{j-1}^{(\nu_{j-1})}(x), \quad B_j^{(\nu_{j-1})} \leftarrow R_{j-1}(x), \quad B_j^{(i)}(x) \leftarrow B_{j-1}^{(i)}(x),\ \forall i \neq \nu_j.$$

as long as $R_j(x) \neq 0$.

An example of the GEA is given in [4, Section II, Example 2].

The EGEA additionally puts out linear factors for each input polynomial.

Theorem 2.6 (Extended Generalized Euclidean Algorithm, EGEA [4]). *Given $s+1$ polynomials $A_0(x)$, $\overline{\mathcal{B}}_0 = \{B_0^{(i)}(x)\}_{i=0}^{s-1}$, where $\overline{\mathcal{B}}_0$ is an element of the set of representatives (4), $A_0(x) \sim B_0^{(\nu_0)}(x)$ and $\deg A_0(x) \geq \deg B_0^{(\nu_0)}(x)$. The EGEA repeatedly applies the generalized division (Algorithm 2) to obtain (7) and calculate polynomials $V_j(x)$ and $\{U_j^{(i)}(x)\}_{i=0}^{s-1}$ such that for each $j = 0, 1 \ldots$*

$$R_j(x) = V_j(x^s) \cdot A_0(x) + \sum_{i=0}^{s-1} U_j^{(i)}(x^s) \cdot B_0^{(i)}(x). \tag{8}$$

Here, we do not explain in detail how $V_j(x^s)$ and $\{U_j^{(i)}(x^s)\}_{i=0}^{s-1}$ are calculated, but we give the connection to the notation from [4]. Let $d_j^{(i)}(x^s)$, $u_j(x^s)$ be the polynomials from [4] that are calculated by [4, Equations (15.2)–(18.2)]. Then, the connection to our notation is as follows:

$$V_j(x^s) \stackrel{\text{def}}{=} d_j^{(\nu_0)}(x^s),\ U_j^{(\nu_0)}(x^s) \stackrel{\text{def}}{=} u_j(x^s),\ U_j^{(i)}(x^s) \stackrel{\text{def}}{=} d_j^{(i)}(x^s), \forall i \neq \nu_0. \tag{9}$$

In order to use the EGEA for multi–sequence shift–register synthesis of s sequences of equal length n, let the input of the EGEA be

$$A_0(x) \leftarrow x^{sn+\nu_0},\ B_0^{(\nu_0)}(x) \leftarrow S(x),\ \{B_0^{(i)}(x)\}_{i=0,i\neq\nu_0}^{s-1} \leftarrow \{x^{sn+i}\}_{i=0,i\neq\nu_0}^{s-1}, \tag{10}$$

where $S(x)$ is calculated by (2). Let the EGEA run up to the *first* remainder where $\deg R_j(x) < U_j^{(\nu_j)}(x^s)$. According to [4, Lemma 4], $\sigma(x) \stackrel{\text{def}}{=} a \cdot U_j^{(\nu_j)}(x)$ is the shortest LFSR that generates each of the s sequences $\mathbf{S}^{(i)}$, $i = 0, \ldots, s-1$ where a is an arbitrary constant. Algorithm 3 shows the EGEA with the breaking condition for multi–sequence shift–register synthesis (Line 1). An example of the EGEA applied for multi–sequence shift–register synthesis is shown in [4, Section III, Example 3].

Algorithm 3: Extended Generalized Euclidean Algorithm (`EGEA`)

Input: $A_0(x)$, $\{B^{(i)}(x)\}_{i=0}^{s-1}$ with $A_0(x) \sim B^{(\nu_0)}(x)$, $\deg A_0(x) \geq \deg B^{(\nu_0)}(x)$

Initialize: $R_0(x) \leftarrow A_0(x), U_0^{(\nu_0)}(x^s) \leftarrow 1, j \leftarrow 0$

1 **while** $\deg U_j^{(\nu_j)}(x^s) \leq \deg R_j(x)$ **do** `// breaking condition`

2 Calculate ν_j such that $R_j(x) \sim B^{(\nu_j)}(x)$

3 $R_{j+1}(x), \{Q^{(i)}(x)\}_{i=0}^{s-1} \leftarrow$ `GenDA` $\left(R_j(x), \{B^{(i)}(x)\}_{i=0}^{s-1}\right)$ `// gen. division`

4 Calculate $V_j(x^s)$ and $\{U_j^{(i)}(x^s)\}_{i=0}^{s-1}$ by (9) and [4, Equations (15.2)–(18.2)]

5 $j \leftarrow j+1$

Output: $V_j(x^s)$, $\{U_j^{(i)}(x^s)\}_{i=0}^{s-1}$

3. Fast extended generalized Euclidean algorithm

3.1. Idea. In order to apply the DC strategy, we want to break the EGEA into two halves. To reduce the complexity, we have to find an efficient way to calculate the modified division (Algorithm 1) and the generalized division (Algorithm 2). Together with a DC strategy of the EGEA (Algorithm 3), the overall complexity can be reduced.

The modified division algorithm (Algorithm 1) of two polynomials $A(x), B(x)$ with $A(x) \sim B(x)$ and $\deg A(x) \geq B(x)$ consists of the first steps of a usual division of the same two polynomials. The iterations of the modified division either stop if the remainder is in another congruence class or if the remainder is still in the same congruence class and $\deg R(x) < \deg B(x)$. For an efficient calculation of the modified division we can truncate the input polynomials as shown in the following. A similar strategy was used by Aho and Hopcroft [1] to accelerate the EA and by Blahut [2] to accelerate the EEA. Let us rewrite the input polynomials $A(x), B(x)$ by:

$$\begin{aligned} A(x) &= \widetilde{A}(x) \cdot x^k + \widehat{A}(x), \\ B(x) &= \widetilde{B}(x) \cdot x^k + \widehat{B}(x), \end{aligned} \tag{11}$$

where both $\deg \widehat{A}(x), \deg \widehat{B}(x) < k$ for some k satisfying

$$k \leq 2 \deg B(x) - \deg A(x). \tag{12}$$

If the modified division is applied to both pairs, $A(x), B(x)$ and $\widetilde{A}(x), \widetilde{B}(x)$, then the quotients and some leading coefficients of the remainders coincide.

Theorem 3.1 (Truncation and the Modified Division)**.** *Given two pairs of polynomials* $A(x) \sim B(x)$ *in* $\mathbb{F}_q[x]$*, with* $\deg A(x) \geq \deg B(x)$ *and* $\widetilde{A}(x) \sim \widetilde{B}(x)$*, with* $\deg \widetilde{A}(x) \geq \deg \widetilde{B}(x)$*. Let us rewrite these polynomials as in* (11) *and let* k *satisfy* (12)*. Let*

$$A(x) = Q(x^s) \cdot B(x) + R(x),$$
$$\widetilde{A}(x) = \widetilde{Q}(x^s) \cdot \widetilde{B}(x) + \widetilde{R}(x)$$

be the results of the modified divisions (Algorithm 1). Then, the quotients and remainders satisfy

$$\begin{aligned} Q(x^s) &= \widetilde{Q}(x^s), \\ R(x) &= \widetilde{R}(x) \cdot x^k + \widehat{R}(x), \end{aligned} \tag{13}$$

where $\deg \widehat{R}(x) < k + \deg A(x) - \deg B(x)$*.*

Proof. In [2, Theorem 10.7.1], Blahut shows that (13) is fulfilled when the *usual* division of two polynomials and their truncated polynomials is calculated. The modified division is a usual division that might stop earlier than the usual division, but never later. Therefore, at most the same number of coefficients of the input polynomials influence the result compared to a usual division. Hence, we can truncate (at least) the same number of coefficients as for a usual division and [2, Theorem 10.7.1] can also be applied for the modified division. □

From Theorem 3.1, if $k \leq 2\deg B(x) - \deg A(x)$, the quotient polynomial of the modified division $Q(x^s)$ does not depend on the k lower coefficients of $A(x)$ and $B(x)$, where $A(x) \sim B(x)$. Similarly, these k lower coefficients affect only the $k + \deg A(x) - \deg B(x)$ lowest coefficients of the remainder $R(x)$.

The following lemma proves that the missing part of the remainder $R(x)$ has no impact on half of the iterations of the generalized division (and hence of the EGEA).

Lemma 3.2 (Truncation and the Generalized Division)**.** *Let* $A(x), \{B^{(i)}(x)\}_{i=0}^{s-1}$ *be* $s+1$ *polynomials with* $\deg A(x) \geq \deg B^{(i)}(x)$*,* $\forall i$*. Let*

$$P_j(x) \stackrel{\text{def}}{=} \sum_{i=0}^{s-1} Q_j^{(i)}(x^s) \cdot B^{(i)}(x),$$
$$\widetilde{P}_j(x) \stackrel{\text{def}}{=} \sum_{i=0}^{s-1} \widetilde{Q}_j^{(i)}(x^s) \cdot \widetilde{B}^{(i)}(x),$$

where $Q_j^{(i)}(x^s)$ *and* $\widetilde{Q}_j^{(i)}(x^s)$ *are the quotients of the generalized division in step* j *for the input polynomials* $A(x), B^{(i)}(x)$ *and* $\widetilde{A}(x), \widetilde{B}^{(i)}(x)$*, respectively. Then,*

$$P_j(x) = \widetilde{P}_j(x) \tag{14}$$

for each j *where* $\deg \widetilde{R}_j(x) \geq (\deg A(x) - k)/2$ *and* $\widetilde{R}_j(x)$ *is the remainder in the generalized division for the input polynomials* $\widetilde{A}(x), \{\widetilde{B}^{(i)}(x)\}_{i=0}^{s-1}$ *in step* j*.*

Proof. From [2, Theorem 10.7.3], we know that if $\deg \widetilde{R}_j(x) \geq (\deg A(x) - k)/2$, then the quotients of the usual divisions of both inputs coincide. Since the modified division stops not later than a usual division, this also holds for the quotients of the modified division. Hence, the results for the generalized divisions are the same and $P_j(x) = \widetilde{P}_j(x)$. □

Hence, about half of the iterations can be calculated correctly without knowing this part. By means of this fact, a fast recursive generalized division algorithm and fast recursive EGEA can be designed that both use truncated polynomials.

3.2. Algorithms. We use the DC strategy combined with a truncation of the polynomials to design a fast EGEA.

To apply the DC strategy, we split the EGEA into two halves. The first half is given in Algorithm 5 (`FH-EGEA`). As typical for DC algorithms, Algorithm 5 contains two recursive calls with the truncated polynomials (Lines 7 and 21) and a generalized division in between. The if–conditions are necessary to check if a truncation is possible. Finally, in Line 24, we calculate the remainder. Note that we use the symbol _ to denote that the output of an algorithm is not used further (see Lines 7, 21 of Algorithm 5).

Algorithm 4 (`F-EGEA`) is the complete fast EGEA that consists of two halves. The first degree condition in Line 1 decides whether the problem can immediately be truncated in Algorithm 5 (`FH-EGEA`) or if a usual generalized division is done. If $\deg B^{(\nu)}(x) \le \deg A(x)/2$, the truncation cannot be applied yet, since $k \le 0$ and we call the usual generalized division. However, we do not lose the reduction of complexity, since in this case one normal generalized division divides the size of the problem in half. The second part of Algorithm 4 has the same form as the original problem and hence, Algorithm 4 is called recursively in Line 8.

The algorithm terminates if $\deg U^{(\nu)}(x^s) > \deg A(x)$. Since this is a recursive algorithm, it also needs $V(x)$, $\{U^{(i)}(x)\}_{i=0}^{s-1}$ as input. For the initialization, we hand over $V(x) = 0$, $U^{(\nu)}(x) = 1$, $\{U^{(i)}(x)\}_{i=0,i\neq\nu}^{s-1} = \{0\}_{i=0,i\neq\nu}^{s-1}$. In the recursions, the current values are used. Note that in contrast to the original algorithms from Section 2, the polynomials do not have the index j since it changes during the recursive calls.

Algorithm 4: Fast EGEA (`F-EGEA`)

Input: $A_0(x)$, $\{B^{(i)}(x)\}_{i=0}^{s-1}$ with $A(x) \sim B^{(\nu_0)}(x)$, $\deg A(x) \ge \deg B^{(\nu_0)}(x)$, $V(x^s)$, $\{U^{(i)}(x^s)\}_{i=0}^{s-1}$

Initialize: $A(x) \leftarrow A_0(x)$, $\{Q^{(i)}(x)\}_{i=0}^{s-1} \leftarrow \{0\}_{i=0}^{s-1}$, $k \leftarrow \lfloor \deg(A(x))/2 \rfloor$, ν where $A(x) \sim B^{(\nu)}(x)$

1 **if** $\deg B^{(\nu)}(x) \le k$ ***or*** $\deg A(x) \le \deg B^{(\nu)}(x)$ **then**
2 $\quad A(x), \{Q^{(i)}(x)\}_{i=0}^{s-1} \leftarrow$ `GenDA` $\left(A(x), \{B^{(i)}(x)\}_{i=0}^{s-1}\right)$
3 **else**
4 $\quad A(x), \{Q^{(i)}(x)\}_{i=0}^{s-1} \leftarrow$ `FH-EGEA` $\left(A(x), \{B^{(i)}(x)\}_{i=0}^{s-1}\right)$
5 Calculate $V(x^s)$ and $\{U^{(i)}(x^s)\}_{i=0}^{s-1}$ by (9), [4, Equations (15.2)–(18.2)]
6 Calculate ν where $A(x) \sim B^{(\nu)}(x)$
7 **if** $\deg U^{(\nu)}(x^s) \le \deg A(x)$ **then**
8 $\quad A(x), V(x), \{U^{(i)}(x)\}_{i=0}^{s-1} \leftarrow$ `F-EGEA` $\left(A(x), \{B^{(i)}(x)\}_{i=0}^{s-1}, V(x), \{U^{(i)}(x)\}_{i=0}^{s-1}\right)$

Output: $A(x)$, $V(x)$, $\{U^{(i)}(x)\}_{i=0}^{s-1}$

Remark 1 (Truncation in the Recursions)**.** The choice of $k = \lfloor \frac{1}{2} \deg A(x) \rfloor$ is motivated by the fact that a Divide and Conquer approach requires that the problem is halved. We now explain how this choice of k agrees with Theorem 3.1. In this remark, we index the polynomials with the recursion level, e.g. $A_j(x)$. Let the

current recursion level of Algorithm FH-EGEA be $j = 0$ and $k = \lfloor \frac{1}{2} \deg A_0(x) \rfloor$. Let the first if-condition be fulfilled, i.e., $\deg B_0^{(\nu)}(x) \leq k$ or $\deg A_0(x) \leq \deg B^{(\nu)}(x)$, then $R_0(x) = A_0(x)$ and $\{Q_0^{(i)}(x)\}_{i=0}^{s-1} = \{0\}_{i=0}^{s-1}$ (see Line 2) are returned. One level higher (where $j = 1$), we calculate in Line 8 of Algorithm FH-EGEA $A_1(x) \leftarrow A_1(x)$ and we know from the lower recursion level that $\deg B_0(x) = \deg B_1(x) - \deg A_1(x)/2 \leq \deg A_0(x)/2 = \deg A_1(x)/4$. Hence, $B_1(x) \leq 3/4 \deg A_1(x)$. Using this, $k = \deg A_1(x)/2 \leq 2B_1(x) - \deg A_1(x)$ is true and due to Theorem 3.1, the generalized division GenDA can be applied (see Line 12).

Algorithm 5: Fast Half EGEA (FH-EGEA)

Input: $A_0(x)$, $\{B^{(i)}(x)\}_{i=0}^{s-1}$

Initialize: $A(x) \leftarrow A_0(x)$, $\{Q^{(i)}(x)\}_{i=0}^{s-1} \leftarrow \{0\}_{i=0}^{s-1}$, $k \leftarrow \lfloor \deg(A(x))/2 \rfloor$, ν where $A(x) \sim B^{(\nu)}(x)$

1 **if** $\deg B^{(\nu)}(x) \leq k$ ***or*** $\deg A(x) \leq \deg B^{(\nu)}(x)$ **then** // no trunc. possible
2 $\quad R(x) \leftarrow A(x)$
3 **else**
4 $\quad \widetilde{A}(x) \leftarrow (A(x) - (A(x) \mod x^k)) \cdot x^{-k}$ // truncation
5 $\quad$ **for** $i = 0$ *to* $s - 1$ **do**
6 $\quad\quad \widetilde{B}^{(i)}(x) \leftarrow (B^{(i)}(x) - (B^{(i)}(x) \mod x^k)) \cdot x^{-k}$
7 $\quad$ –, $\{Q^{(i)}(x)\}_{i=0}^{s-1} \leftarrow$ FH-EGEA $\left(\widetilde{A}(x), \{\widetilde{B}^{(i)}(x)\}_{i=0}^{s-1}\right)$ // 1st recursion
8 $\quad A(x) \leftarrow A(x) - \sum_{i=0}^{s-1} Q^{(i)}(x) \cdot \widetilde{B}^{(i)}(x)$
9 $\quad$ Calculate ν such that $A(x) \sim B^{(\nu)}(x)$
10 $\quad k \leftarrow \lfloor \deg(A(x))/2 \rfloor$
11 $\quad$ **if** $\deg A(x) > \deg B^{(\nu)}(x)$ **then**
12 $\quad\quad A(x), \{\widetilde{Q}^{(i)}(x)\}_{i=0}^{s-1} \leftarrow$ GenDA $\left(A(x), \{B^{(i)}(x)\}_{i=0}^{s-1}\right)$ // gen. division
13 $\quad\quad$ **for** $i = 0$ *to* $s - 1$ **do**
14 $\quad\quad\quad Q^{(i)}(x) \leftarrow Q^{(i)}(x) + \widetilde{Q}^{(i)}(x)$
15 $\quad\quad$ Calculate ν such that $A(x) \sim B^{(\nu)}(x)$
16 $\quad\quad k \leftarrow \lfloor \deg(A(x))/2 \rfloor$
17 $\quad\quad$ **if** $\deg A(x) > \deg B^{(\nu)}(x) > k$ **then**
18 $\quad\quad\quad \widetilde{A}(x) \leftarrow (A(x) - (A(x) \mod x^k)) \cdot x^{-k}$ // truncation
19 $\quad\quad\quad$ **for** $i = 0$ *to* $s - 1$ **do**
20 $\quad\quad\quad\quad \widetilde{B}^{(i)}(x) \leftarrow (B^{(i)}(x) - (B^{(i)}(x) \mod x^k))x^{-k}$
21 $\quad\quad\quad$ –, $\{\widetilde{Q}^{(i)}(x)\}_{i=0}^{s-1} \leftarrow$ FH-EGEA $\left(\widetilde{A}(x), \{\widetilde{B}^{(i)}(x)\}_{i=0}^{s-1}\right)$ // 2nd rec.
22 $\quad\quad\quad$ **for** $i = 0$ *to* $s - 1$ **do**
23 $\quad\quad\quad\quad Q^{(i)}(x) \leftarrow Q^{(i)}(x) + \widetilde{Q}^{(i)}(x)$
24 $\quad R(x) \leftarrow A_0(x) - \sum_{i=0}^{s-1} Q^{(i)}(x) \cdot B^{(i)}(x)$ // calculate remainder

Output: $R(x)$, $\{Q^{(i)}(x)\}_{i=0}^{s-1}$

4. Complexity analysis

4.1. Complexity of the EGEA.

Feng and Tzeng did not analyze the complexity

of the EGEA in [4]. Based on the idea of Lipson [10, Chapter 7] for the complexity analysis of the EEA, we give a bound on the time complexity of the EGEA (Definition 2.6).

We count the number of multiplications in the finite field $\mathbb{F}_q$ of order q. Let $N \geq \deg A_0(x) \geq \deg\{B_0^{(i)}(x)\}_{i=0}^{s-1}$. Let us consider the GEA first. In the general case, the GEA runs up to $R_j(x) = 0$ (see (7)). For simplicity, we assume that in every jth step the degree of the polynomial $R_j(x)$ (see (7)) decreases by one. Consider Equation (7) of Definition 2.5. A simple division has the same complexity as a multiplication of two polynomials with some scalar factor. The number of operations $\mathcal{T}_j$ in $\mathbb{F}_q$ for the jth step of the GEA is:

$$\mathcal{T}_j = c_2 \cdot \frac{1}{s} \sum_{i=0}^{s-1} \deg Q_j^{(i)}(x^s) \cdot \deg B_j^{(i)}(x),$$

where the factor $1/s$ comes from the "sparsity" (only every sth coefficient is considered) of the polynomials $Q_j^{(i)}(x^s)$. Summing up over all N iterations and since $\deg B_j^{(i)}(x) \leq N \; \forall i, j$, we obtain:

$$\begin{aligned} \mathcal{T} = \sum_{j=1}^{N} \mathcal{T}_j &\leq c\frac{1}{s} \sum_{j=1}^{N} \sum_{i=0}^{s-1} \deg Q_j^{(i)}(x^s) \cdot \deg B_j^{(i)}(x), \\ &\leq c\frac{1}{s} N \sum_{j=1}^{N} \sum_{i=0}^{s-1} \deg Q_j^{(i)}(x^s). \end{aligned} \tag{15}$$

We know from [4, Equations (8.3), (8.4)] that:

$$\begin{aligned} \deg Q_j^{(\nu_j)}(x^s) &= \deg A_j(x) - \deg B_j^{(\nu_j)}(x), \\ \deg Q_j^{(i)}(x^s) &< \deg A_j(x) - \deg B_j^{(i)}(x) \quad \forall i \neq \nu_j, \end{aligned}$$

where $\deg R_j(x) \sim B_j^{(\nu_j)}(x)$. Hence, we have:

$$\begin{aligned} \sum_{j=1}^{N} \sum_{i=0}^{s-1} \deg Q_j^{(i)}(x^s) &\leq \sum_{j=1}^{N} \sum_{i=0}^{s-1} (\deg A_j(x) - \deg B_j^{(i)}(x)) \\ &\leq sN. \end{aligned}$$

As we assume that the degree of the remainder decreases by one, we obtain $\mathcal{T} \leq cN^2$. The additional calculations to extend the GEA to the EGEA do not affect this bound on the time complexity and we conclude with the following theorem for the EGEA.

Theorem 4.1 (Complexity of EGEA, Algorithm 3)**.** *For $s+1$ polynomials $A_0(x)$, $\{B_0^{(i)}(x)\}_{i=0}^{s-1} \in \mathbb{F}_q[x]$ with degree smaller than or equal to N, the EGEA (Algorithm 3) of Feng and Tzeng has time complexity $\mathcal{O}(N^2)$, assuming that coefficient operations take constant $\mathcal{O}(1)$ time.*

If Algorithm 3 is applied to Problem 2.2 at most n iterations (instead of sn) have to be performed. Therefore, we have the following theorem.

Theorem 4.2 (Complexity of Algorithm 3 applied to Multi–Sequence Shift–Register Synthesis)**.** *Let the $s+1$ polynomials $S(x)$ and $\{x^{sn+i}\}_{i=0}^{s-1}$ in $\mathbb{F}_q[x]$ be given for the fast EGEA as stated in Problem 2.2. Then the EGEA of Feng and Tzeng solves Problem 2.2 with time complexity $\mathcal{O}(sn^2)$.*

This is equivalent to the time complexity of Feng and Tzeng's generalized Berlekamp-Massey [5] approach.

4.2. COMPLEXITY OF THE FAST EGEA. Throughout this section, let $\mathcal{MD}(N)$ denote the complexity of a modified division, where $N = \deg A(x) \geq \deg B(x)$.

Let us first analyze the complexity of the fast half EGEA, Algorithm 5 (`FH-EGEA`). Let $N \geq \deg A_0(x)$ and let $\mathcal{T}_{FHEGEA}(N)$ denote the maximum running time of Algorithm 5. The complexity of the splitting operations and the additions is negligible. The generalized division can be implemented by (at most) s parallel modified divisions, i.e., it requires $c_1 \cdot s \cdot \mathcal{MD}(N)$ operations, for a constant c_1. In addition, Algorithm 5 includes two recursive calls with half size. Hence, $\mathcal{T}_{FHEGEA}(N)$ is upper bounded by:

$$\mathcal{T}_{FHEGEA}(N) \leq 2 \cdot \mathcal{T}_{FHEGEA}\left(\frac{N}{2}\right) + c_1 \cdot s \cdot \mathcal{MD}(N).$$

It is well–known that this linear recurrence relation implies (see e.g. [6, Lemma 8.2]):

$$\mathcal{T}_{FHEGEA}(N) \leq c_2 \cdot s \cdot \mathcal{MD}(N) \log N,$$

for some constant c_2. Thus, Algorithm 5 requires $\mathcal{O}(s \cdot \mathcal{MD}(N) \log N)$ operations in $\mathbb{F}_q$.

Based on the complexity of Algorithm 5, we can analyze the complexity of Algorithm 4 (`F-EGEA`).

Theorem 4.3 (Complexity of fast EGEA, Algorithm 4)**.** *For $s+1$ polynomials $A_0(x), \{B_0^{(i)}(x)\}_{i=0}^{s-1} \in \mathbb{F}_q[x]$ with degree smaller than or equal to N, the fast EGEA (Algorithm 4, `F-EGEA`) has a time complexity $\mathcal{O}(s \cdot \mathcal{MD}(N) \log N) \leq \mathcal{O}(sN \log^2 N)$, assuming that coefficient operations take constant $\mathcal{O}(1)$ time.*

Proof. Let $\mathcal{T}_{FEGEA}(N)$ denote the maximum running time of Algorithm 4. In the first if–condition, the algorithm includes either a generalized division with complexity $s \cdot \mathcal{MD}(N)$ or a call of Algorithm 5 with complexity $\mathcal{O}(s \cdot \mathcal{MD}(N) \log N)$. Afterwards, there are additions with complexity $\mathcal{O}(N)$ to calculated the polynomials $V(x^s)$, $\{U^{(i)}(x^s)\}_{i=0}^{s-1}$ and a recursive call with half the size.

Since $\mathcal{T}_{FHEGEA}(N) = \mathcal{O}(s \cdot \mathcal{MD}(N) \log N) > s \cdot \mathcal{MD}(N) > \mathcal{O}(N)$, $\mathcal{T}_{FEGEA}(N)$ is upper bounded by:

$$\mathcal{T}_{FEGEA}(N) \leq T_{FHEGEA}(N) + \mathcal{T}_{FEGEA}\left(\frac{N}{2}\right).$$

Using [6, Lemma 8.2], this inequality can be upper bounded by:

$$\mathcal{T}_{FEGEA}(N) \leq c_3 \cdot \mathcal{T}_{FHEGEA}(N) \leq c_4 \cdot s \cdot \mathcal{MD}(N) \log N,$$

and Algorithm 4 (`F-EGEA`) requires $\mathcal{O}(s \cdot \mathcal{MD}(N) \log N)$ operations in $\mathbb{F}_q$ if $N = \deg A_0(x)$. With worst case assumptions, $\mathcal{MD}(N)$ has the same complexity as a usual division of two polynomials of length N. Such a usual division can be done with the complexity of multiplying two polynomials of length N [6]. Hence, using a fast Fourier transform for polynomial multiplication, we have $\mathcal{MD}(N) \leq \mathcal{O}(N \log N)$ and

$$\mathcal{T}_{FEGEA}(N) \leq \mathcal{O}(sN \log^2 N).$$

□

For multi–sequence shift–register synthesis of equal length, each sequence has length n and the overall length of the sequences is $N = sn$. With worst case assumptions, we have $\mathcal{MD}(N) = \mathcal{MD}(sn) \leq \mathcal{O}(N \log N) = \mathcal{O}(sn \log(sn))$.

Thus, the complexity of the fast EGEA is given by the following theorem.

Theorem 4.4 (Complexity of fast EGEA applied to Multi–Sequence Shift–Register Synthesis)**.** *Let the $s+1$ polynomials $S(x)$ and $\{x^{sn+i}\}_{i=0}^{s-1}$ in $\mathbb{F}_q[x]$ be given for the fast EGEA as stated in Problem 2.2. Then the fast EGEA (Algorithm 4, `F-EGEA`) with the input polynomials as in (10), solves Problem 2.2 with time complexity*

$$\mathcal{T}_{FEGEA}(N) \leq \mathcal{O}(sN \log^2 N) \leq \mathcal{O}(s^2 n \log^2(sn))$$

This complexity is subquadratic, i.e., for large sequence lengths (and as usual, small numbers of s), the complexity is reduced by our accelerated algorithms.

5. Conclusion

We investigated the multi–sequence shift–register synthesis problem for sequences of equal length, which can be solved by Feng and Tzeng's EGEA. The complexity of the EGEA was analyzed and we reduced the time complexity to subquadratic complexity by application of a DC strategy.

Acknowledgement

The authors thank Vladimir Sidorenko and the reviewers for their valuable comments and suggestions.

References

[1] A. V. Aho and J. E. Hopcroft, "The Design and Analysis of Computer Algorithms," Addison-Wesley Longman Publishing Co., 1974.
[2] R. E. Blahut, "Fast Algorithms for Digital Signal Processing," Addison-Wesley, 1985.
[3] D. Bleichenbacher, A. Kiayias and M. Yung, *Decoding interleaved Reed–Solomon codes over noisy channels*, Theor. Comput. Sci., **379** (2007), 348–360.
[4] G. L. Feng and K. K. Tzeng, *A generalized Euclidean algorithm for multisequence shift–register synthesis*, IEEE Trans. Inform. Theory, **35** (1989), 584–594.
[5] G. L. Feng and K. K. Tzeng, *A generalization of the Berlekamp–Massey algorithm for multisequence shift–register synthesis with applications to decoding cyclic codes*, IEEE Trans. Inform. Theory, **37** (1991), 1274–1287.
[6] J. Gathen and J. Gerhard, "Modern Computer Algebra", Cambridge University Press, 2003.
[7] C. Hartmann, *Decoding beyond the BCH bound*, IEEE Trans. Inform. Theory, **18** (1972), 441–444.
[8] V. Y. Krachkovsky, *Reed–Solomon codes for correcting phased error bursts*, IEEE Trans. Inform. Theory, **49** (2003), 2975–2984.
[9] V. Y. Krachkovsky and Y. X. Lee, *Decoding for iterative Reed–Solomon coding schemes*, IEEE Trans. Magnetics, **33** (1997), 2740–2742.
[10] J. D. Lipson, "Elements of Algebra and Algebraic Computing," Addison-Wesley Educational Publishers Inc, 1981.
[11] C. Roos, *A generalization of the BCH bound for cyclic codes, including the Hartmann-Tzeng bound*, J. Combin. Theory Ser. A, **33** (1982), 229–232.
[12] G. Schmidt, V. R. Sidorenko and M. Bossert, *Syndrome decoding of Reed–Solomon codes beyond half the minimum distance based on shift–register synthesis*, IEEE Trans. Inform. Theory, **56** (2010), 5245–5252.
[13] Y. Sugiyama, M. Kasahara, S. Hirasawa and T. Namekawa, *A method for solving key equation for decoding Goppa codes*, Inform. Control, **27** (1975), 87–99.
[14] L. Wang, *Euclidean modules and multisequence synthesis*, in "Applied Algebra, Algebraic Algorithms and Error-Correcting Codes" (eds. S. Boztaş and I.E. Shparlinski), Springer, Berlin, Heidelberg, (2001), 239–248.

[15] A. Zeh and W. Li, *Decoding Reed–Solomon codes up to the Sudan radius with the Euclidean algorithm*, in "Proceedings of the 2010 International Symposium on Information Theory and its Applications (ISITA)," Taichung, Taiwan, (2010), 986–990.

Received November 2010; revised June 2011.

E-mail address: Alexander.Zeh@uni-ulm.de
E-mail address: Antonia.Wachter@uni-ulm.de