SUBRESULTANTS REVISITED

Extended Abstract

Joachim von zur Gathen and Thomas Lücking

FB Mathematik-Informatik, Universität Paderborn
33095 Paderborn, Germany
{gathen,luck}@upb.de

1 Introduction

1.1 Historical context

The *Euclidean Algorithm* was first documented by Euclid (320–275 BC). Knuth (1981), p. 318, writes: *“We might call it the granddaddy of all algorithms, because it is the oldest nontrivial algorithm that has survived to the present day.”* It performs division with remainder repeatedly until the remainder becomes zero. With inputs 13 and 9 it performs the following:

$$\begin{aligned} 13 &= 1\cdot 9+4,\\ 9 &= 2\cdot 4+\boxed{1},\\ 4 &= 4\cdot 1+0. \end{aligned}$$

This allows us to compute the *greatest common divisor (gcd)* of two integers as the last non-vanishing remainder. In the example, the gcd of 13 and 9 is computed as 1.

At the end of the 17th century the concept of polynomials was evolving. Researchers were interested in finding the common roots of two polynomials f and g. One question was whether it is possible to apply the Euclidean Algorithm to f and g. In 1707 Newton solved this problem and showed that this always works in $\mathbb{Q}[x]$.

$$\begin{aligned} x^3+2x^2-x-2 &= (\frac{1}{2}x+\frac{3}{2})(2x^2-2x-4)+\boxed{4x+4}\\ 2x^2-2x-4 &= (\frac{1}{2}x-1)(4x+4)+0. \end{aligned}$$

In this example $f=x^3+2x^2-x-2$ and $g=2x^2-2x-4$ have a greatest common divisor $4x+4$, and therefore the only common root is -1. In a certain sense the Euclidean Algorithm computes all common roots. If you only want to know whether f and g have at least *one* common root, then the whole Euclidean

Algorithm has to be executed. Thus the next goal was to find an indicator for common roots without using any division with remainder.

The key to success was found in 1748 by Euler, and later by Bézout. They defined the *resultant* of f and g as the smallest polynomial in the coefficients of f and g that vanishes if and only if f and g have a common root. In 1764 Bézout was the first to find a matrix whose determinant is the resultant. The entries of this *Bézout matrix* are quadratic functions of the coefficients of f and g. Today we use the matrix discovered by Sylvester in 1840, known as the *Sylvester matrix.* Its entries are simply coefficients of the polynomials f and g. Sylvester generalized his definition and introduced what we now call *subresultants* as determinants of certain submatrices of the Sylvester matrix. They are nonzero if and only if the corresponding degree appears as a degree of a remainder of the Euclidean Algorithm.

These indicators, in particular the resultant, also work for polynomials in $\mathbb{Z}[x]$. So the question came up whether it is possible to apply the Euclidean Algorithm to f and g in $\mathbb{Z}[x]$ without leaving $\mathbb{Z}[x]$. The answer is no, as illustrated in the example above, since division with remainder is not always defined in $\mathbb{Z}[x]$, although the gcd exists. In the example it is $x + 1$.

However, in 1836 Jacobi found a way out. He introduced *pseudo-division*: he multiplied f with a certain power of the leading coefficient of g before performing the division with remainder. This is always possible in $\mathbb{Z}[x]$. So using pseudo-division instead of division with remainder in every step in the Euclidean Algorithm yields an algorithm with all intermediate results in $\mathbb{Z}[x]$.

About 40 years later Kronecker did research on the *Laurent series* in x^{-1} of g/f for two polynomials f and g. He considered the determinants of a matrix whose entries are the coefficients of the Laurent series of g/f. He obtained the same results as Sylvester, namely that these determinants are nonzero if and only if the corresponding degree appears in the degree sequence of the Euclidean Algorithm. Furthermore Kronecker gave a direct way to compute low degree polynomials s, t and r with $sf + tg = r$ via determinants of matrices derived again from the Laurant series of g/f, and showed that these polynomials are essentially the only ones. He also proved that the polynomial r, if nonzero, agrees with a remainder in the Euclidean Algorithm, up to a constant multiple. This was the first occurrence of *polynomial subresultants.*

In the middle of our century, again 70 years later, the realization of computers made it possible to perform more and more complicated algorithms faster and faster. However, using *pseudo-division* in every step of the Euclidean Algorithm causes *exponential* coefficient growth. This was suspected in the late 1960's. Collins (1967), p. 139 writes: *"Thus, for the Euclidean algorithm, the lengths of the coefficients increases exponentially."* In Brown & Traub (1971) we find: *"Although the Euclidean PRS algorithm is easy to state, it is thoroughly impractical since the coefficients grow exponentially."* An exponential *upper* bound is in Knuth (1981), p. 414: *"Thus the upper bound* [...] *would be approximately* $N^{0.5(2.414)^n}$*, and experiments show that the simple algorithm does in fact have this behavior; the number of digits in the coefficients grows* *exponentially at each step!*". However, we did not find a proof of an exponential *lower* bound; our bound in Theorem 7.3 seems to be new.

One way out of this exponential trap is to make every intermediate result *primitive*, that is, to divide the remainders by the greatest common divisors of their coefficients, the so-called *content*. However, computing the contents seemed to be very expensive since in the worst case the gcd of *all* coefficients has to be computed. So the scientists tried to find divisors of the contents without using any gcd computation. Around 1970, first Collins and then Brown & Traub reinvented the *polynomial subresultants* as determinants of a certain variant of the Sylvester matrix. Habicht had also defined them independently in 1948. Collins and Brown & Traub showed that they agree with the remainders of the Euclidean Algorithm up to a constant factor. They gave simple formulas to compute this factor and introduced the concept of *polynomial remainder sequences (PRS)*, generalizing the concept of Jacobi. The final result is the *subresultant PRS* that features linear coefficient growth with intermediate results in $\mathbb{Z}[x]$.

Since then two further concepts have come up. On the one hand the *fast EEA* allows to compute an arbitrary intermediate line in the Euclidean Scheme directly. Using the fast $O(n \log n \log \log n)$ multiplication algorithm of Schönhage and Strassen, the time for a gcd reduces from $O(n^2)$ to $O(n \log^2 n \log \log n)$ field operations (see Strassen (1983)). On the other hand, the *modular EEA* is very efficient. These two topics are not considered in this thesis; for further information we refer to von zur Gathen & Gerhard (1999), Chapters 6 and 11.

1.2 Outline

After introducing the notation and some well-known facts in Section 2, we start with an overview and comparison of various definitions of subresultants in Section 3. Mulders (1997) describes an error in software implementations of an integration algorithm which was due to the confusion caused by the these various definitions. It turns out that there are essentially two different ways of defining them: the scalar and the polynomial subresultants. Furthermore we show their relation with the help of the Euclidean Algorithm. In the remainder of this work we will mainly consider the scalar subresultants.

In Section 4 we give a formal definition of polynomial remainder sequences and derive the most famous ones as special cases of our general notion. The relation between polynomial remainder sequences and subresultants is exhibited in the Fundamental Theorem 5.1 in Section 5. It unifies many results in the literature on various types of PRS which can be derived as corollaries from this theorem. In Section 6 we apply it to the various definitions of polynomial remainder sequences already introduced. This yields a collection of results from Collins (1966, 1967, 1971, 1973), Brown (1971, 1978), Brown & Traub (1971), Lickteig & Roy (1997) and von zur Gathen & Gerhard (1999). Lickteig & Roy (1997) found a recursion formula for polynomial subresultants not covered by the Fundamental Theorem. We translate it into a formula for scalar subresultants and use it to finally solve an open question in Brown (1971), p. 486. In Section 7 we analyse the coefficient growth and the running time of the various PRS.

Finally in Section 8 we report on implementations of the various polynomial remainder sequences and compare their running times. It turns out that computing contents is quite fast for random inputs, and that the primitive PRS behaves much better than expected.

Much of this Extended Abstract is based on the existing literature. The following results are new:

- rigorous and general definition of division rules and PRS,
- proof that all constant multipliers in the subresultant PRS for polynomials over an integral domain R are also in R,
- exponential lower bound for the running time of the pseudo PRS (algorithm).

2 Foundations

In this chapter we introduce the basic algebraic notions. We refer to von zur Gathen & Gerhard (1999), Sections 2.2 and 25.5, for the notation and fundamental facts about greatest common divisors and determinants. More information on these topics is in Hungerford (1990).

2.1 Polynomials

Let R be a ring. In what follows, this always means a commutative ring with 1.

A basic tool in computer algebra is *division with remainder.* For given polynomials f and g in $R[x]$ of degrees n and m, respectively, the task is to find polynomials q and r in $R[x]$ with

$$f = qg + r \text{ and } \deg r < \deg g. \tag{2.1}$$

Unfortunately such q and r do not always exist.

Example 2.2. It is not possible to divide x^2 by $2x + 3$ with remainder in $\mathbb{Z}[x]$ because $x^2 = (ux + v)(2x + 3) + r$ with $u, v, r \in \mathbb{Q}$ has the unique solution $u = 1/2$, $v = 0$ and $r = -3/2$, which is not over $\mathbb{Z}$.

If defined and unique we call $q = \mathrm{quo}(f, g)$ the *quotient* and $r = \mathrm{rem}(f, g)$ the *remainder.* A ring with a length function (like the degree of polynomials) and where division with remainder is always defined is a *Euclidean domain.* $R[x]$ is a Euclidean domain if and only if R is a field. Moreover a solution of (2.1) is not necessarily unique if the leading coefficient $\mathrm{lc}(g)$ of g is a zero divisor.

Example 2.3. Let $R = \mathbb{Z}_8$ and consider $f = 4x^2 + 2x$ and $g = 2x + 1$. With

$$\begin{aligned} q_1 &= 2x, & r_1 &= 0 \\ q_2 &= 2x + 4, & r_2 &= 4 \end{aligned}$$

we obtain

$$\begin{aligned} q_1 g + r_1 &= 2x(2x + 1) + 0 = 4x^2 + 2x = f, \\ q_2 g + r_2 &= (2x + 4)(2x + 1) + 4 = 4x^2 + 10x + 8 = 4x^2 + 2x = f. \end{aligned}$$

Thus we have two distinct solutions (q_1, r_1) and (q_2, r_2) of (2.1).

A way to get solutions for all commutative rings is the *general pseudo-division* which allows multiplication of f by a ring element α:

$$\alpha f = qg + r, \ \deg r < \deg g. \tag{2.4}$$

If $\alpha = g_m^{n-m+1}$, then this is the *(classical) pseudo-division.* If $\mathrm{lc}(g)$ is not a zero divisor, then (2.4) always has a unique solution in $R[x]$. We call $q = \mathrm{pquo}(f, g)$ the *pseudo-quotient* and $r = \mathrm{prem}(f, g)$ the *pseudo-remainder.*

Example 2.2 continued. For x^2 and $2x + 3$ we get the pseudo-division

$$2^2 \cdot x^2 = (2x - 3)(2x + 3) + 9$$

A simple computation shows that we cannot choose $\alpha = 2$.

Lemma 2.5.

(i) Pseudo-division always yields a solution of (2.4) in $R[x]$.
(ii) If $\mathrm{lc}(g)$ *is not a zero divisor, then any solution of (2.4) has* $\deg q = n - m$.

Lemma 2.6. *The solution* (q, r) *of (2.4) is uniquely determined if and only if* $\mathrm{lc}(g)$ *is not a zero-divisor.*

Let R be a unique factorization domain. We then have $\gcd(f, g) \in R$ for $f, g \in R[x]$, and the *content* $\mathrm{cont}(f) = \gcd(f_0, \dots, f_n) \in R$ of $f = \sum_{0 \le j \le n} f_j x^j$. The polynomial is *primitive* if $\mathrm{cont}(f)$ is a unit. The *primitive part* $\mathrm{pp}(f)$ is defined by $f = \mathrm{cont}(f) \cdot \mathrm{pp}(f)$. Note that $\mathrm{pp}(f)$ is a primitive polynomial.

The *Euclidean Algorithm* computes the gcd of two polynomials by iterating the division with remainder:

$$r_{i-1} = q_i r_i + r_{i+1}. \tag{2.7}$$

3 Various notions of subresultants

3.1 The Sylvester matrix

The various definitions of the subresultant are based on the *Sylvester matrix.* Therefore we first take a look at the historical motivation for this special matrix. Our goal is to decide whether two polynomials $f = \sum_{0 \le j \le n} f_j x^j$ and $g = \sum_{0 \le j \le m} g_j x^j \in R[x]$ of degree $n \ge m > 0$ over a commutative ring R in the indeterminate x have a common root. To find an answer for this question, Euler (1748) and Bézout (1764) introduced the *(classical) resultant* that vanishes if and only if this is true. Although Bézout also succeeded in finding a matrix whose determinant is equal to the resultant, today called *Bézout matrix*, we will follow the elegant derivation in Sylvester (1840). The two linear equations

$$\begin{aligned} f_n x_n + f_{n-1} x_{n-1} + \cdots + f_1 x_1 + f_0 x_0 &= 0 \\ g_m x_m + g_{m-1} x_{m-1} + \cdots + g_1 x_1 + g_0 x_0 &= 0 \end{aligned}$$

in the indeterminates $x_0, \dots, x_n$ are satisfied if $x_j = \alpha^j$ for all j, where α is a common root of f and g. For $n > 1$ there are many more solutions of these two linear equations in many variables, but Sylvester eliminates them by adding the $(m-1)+(n-1)$ linear equations that correspond to the following additional conditions:

$$\begin{aligned} xf(x) &= 0, \dots, x^{m-1}f(x) = 0, \\ xg(x) &= 0, \dots, x^{n-1}g(x) = 0. \end{aligned}$$

These equations give a total of $n+m$ linear relations among the variables $x_{m+n-1}, \cdots, x_0$:

$$\begin{array}{lllllllll} f_n x_{m+n-1} & + & \cdots & + & f_0 x_{m-1} & & & & = 0 \\ & & \vdots & & & & & & \\ & & f_n x_n & + & f_{n-1}x_{n-1} & + & \cdots & + f_0 x_0 & = 0 \\ g_m x_{m+n-1} & + & \cdots & + & g_0 x_{n-1} & & & & = 0 \\ & & \vdots & & & & & & \\ & & g_m x_m & + & g_{m-1}x_{m-1} & + & \cdots & + g_0 x_0 & = 0 \end{array}$$

Clearly $x_j = \alpha^j$ gives a solution for any common root α of f and g, but the point is that (essentially) the converse also holds: a solution of the linear equations gives a common root (or factor). The $(n+m) \times (n+m)$ matrix, consisting of coefficients of f and g, that belongs to this system of linear equations is often called *Sylvester matrix*. In the sequel we follow von zur Gathen & Gerhard (1999), Section 6.3, p. 144, and take its transpose.

Definition 3.1. *Let R be a commutative ring and let $f = \sum_{0 \le j \le n} f_j x^j$ and $g = \sum_{0 \le j \le m} g_j x^j \in R[x]$ be polynomials of degree $n \ge m > 0$, respectively. Then the $(n+m) \times (n+m)$ matrix*

$$\mathrm{Syl}(f,g) = \underbrace{\left(\begin{array}{ccccccc} f_n & & & g_m & & & \\ f_{n-1} & f_n & & g_{m-1} & g_m & & \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \\ \vdots & \vdots & f_n & g_1 & \vdots & \ddots & \\ \vdots & \vdots & f_{n-1} & g_0 & \vdots & & \ddots \\ \vdots & \vdots & \vdots & & g_0 & & g_m \\ f_0 & \vdots & \vdots & & & \ddots & \vdots \\ & f_0 & \vdots & & & \ddots & \vdots \\ & & \ddots \vdots & & & \ddots & \vdots \\ & & f_0 & & & & g_0 \end{array}\right)}_{m \qquad\qquad\qquad n}$$

is called the Sylvester matrix of f and g.

Remark 3.2. Multiplying the $(n+m-j)$th row by x^j and adding it to the last row for $1 \le j < n+m$, we get the $(n+m) \times (n+m)$ matrix S^*. Thus $\det(\mathrm{Syl}(f,g)) = \det(\mathrm{Syl}^*(f,g))$.

More details on resultants can be found in Biermann (1891), Gordan (1885) and Haskell (1892). Computations for both the univariate and multivariate case are discussed in Collins (1971).

3.2 The scalar subresultant

We are interested in finding out which degrees appear in the degree sequence of the intermediate results in the Euclidean Algorithm. Below we will see that the scalar subresultants provide a solution to this problem.

Definition 3.3. *Let R be a commutative ring and $f = \sum_{0 \le j \le n} f_j x^j$ and $g = \sum_{0 \le j \le m} g_j x^j \in R[x]$ polynomials of degree $n \ge m > 0$, respectively. The determinant $\sigma_k(f,g) \in R$ of the $(m+n-2k) \times (m+n-2k)$ matrix*

$$S_k(f,g) = \left(\underbrace{\begin{array}{ccc}
f_n & & \\
f_{n-1} & f_n & \\
\vdots & \ddots & \\
f_{n-m+k+1} & \cdots\cdots & f_n \\
\vdots & & \vdots \\
f_{k+1} & \cdots\cdots & f_m \\
\vdots & & \vdots \\
\vdots & & \vdots \\
f_{2k-m+1} & \cdots\cdots & f_k
\end{array}}_{m-k}
\underbrace{\begin{array}{cccc}
g_m & & & \\
g_{m-1} & g_m & & \\
\vdots & \ddots & & \\
g_{k+1} & \cdots\cdots & g_m & \\
\vdots & & & \ddots \\
g_{m-n+k+1} & \cdots\cdots & \cdots\cdots & g_m \\
\vdots & & & \vdots \\
\vdots & & & \vdots \\
g_{2k-n+1} & \cdots\cdots & \cdots\cdots & g_k
\end{array}}_{n-k} \right)$$

*is called the k***th (scalar) subresultant of** *f* **and** *g. By convention an f_j or g_j with $j < 0$ is zero. If f and g are clear from the context, then we write S_k and σ_k for short instead of $S_k(f,g)$ and $\sigma_k(f,g)$.*

Sylvester (1840) already contains an explicit description of the (scalar) subresultants. In Habicht (1948), p. 104, σ_k is called *Nebenresultante (minor resultant)* for polynomials f and g of degrees n and $n-1$. The definition is also in von zur Gathen (1984) and is used in von zur Gathen & Gerhard (1999), Section 6.10, p. 169.

Remark 3.4.

(i) $S_0 = \mathrm{Syl}(f,g)$ and therefore $\sigma_0 = \det(S_0)$ is the *resultant*.
(ii) $\sigma_m = g_m^{n-m}$.

(iii) S_k is the matrix obtained from the Sylvester matrix by deleting the last $2k$ rows and the last k columns with coefficients of f, and the last columns with coefficients of g.
(iv) S_k is a submatrix of S_i if $k \geq i$.

3.3 The polynomial subresultant

We now introduce two slightly different definitions of polynomial subresultants. The first one is from Collins (1967), p. 129, and the second one is from Brown & Traub (1971), p. 507 and also in Zippel (1993), Chapter 9.3, p. 150. They yield polynomials that are related to the intermediate results in the Euclidean Algorithm.

Definition 3.5. *Let R be a commutative ring, and $f = \sum_{0\leq j\leq n} f_j x^j$ and $g = \sum_{0\leq j\leq m} g_j x^j \in R[x]$ polynomials of degree $n \geq m > 0$. Let $M_{ik} = M_{ik}(f,g)$ be the $(n+m-2k)\times(n+m-2k)$ submatrix of $\mathrm{Syl}(f,g)$ obtained by deleting the last k of the m columns of coefficients of f, the last k of the n columns of coefficients of g and the last $2k+1$ rows except row $(n+m-i-k)$, for $0 \leq k \leq m$ and $0 \leq i \leq n$. The polynomial $R_k(f,g) = \sum_{0\leq i\leq n} \det(M_{ik})x^i \in R[x]$ is called the* **kth polynomial subresultant of** f **and** g. *In fact Collins (1967) considered the transposed matrices. If f and g are clear from the context, then we write R_k for short instead of $R_k(f,g)$. Note that $\det(M_{ik}) = 0$ if $i > k$ since then the last row of M_{ik} is identical to the $(n+m-i-k)$th row. Thus $R_k = \sum_{0\leq i\leq k} \det(M_{ik})x^i$.*

Remark 3.6.

(i) $M_{00} = \mathrm{Syl}(f,g)$ and therefore $R_0 = \det(M_{00})$ is the resultant.
(ii) Remark 3.4(i) implies $\sigma_0 = R_0$.

Definition 3.7. *Let R be a commutative ring and $f = \sum_{0\leq j\leq n} f_j x^j$ and $g = \sum_{0\leq j\leq m} g_j x^j \in R[x]$ polynomials of degree $n \geq m > 0$. We consider the determinant $Z_k(f,g) = \det(M_k^*) \in R[x]$ of the $(n+m-2k)\times(n+m-2k)$ matrix M_k^* obtained from M_{ik} by replacing the last row with $(x^{m-k-1}f, \cdots, f, x^{n-k-1}g, \cdots, g)$.*

Table 1 gives an overview of the literature concerning these notions. There is a much larger body of work about the special case of the resultant, which we do not quote here.

3.4 Comparison of the various definitions

As in Brown & Traub (1971), p. 508, and Geddes *et al.* (1992), Section 7.3, p. 290, we first have the following theorem which shows that the definitions in Collins (1967) and Brown & Traub (1971) describe the same polynomial.

Theorem 3.8.

(i) If $\sigma_k(f,g) \neq 0$, then $\sigma_k(f,g)$ is the leading coefficient of $R_k(f,g)$. Otherwise, $\deg R_k(f,g) < k$.

Definition	**Authors**
$\sigma_k(f,g) = \det(S_k) \in R$	Sylvester (1840), Habicht (1948)
	von zur Gathen (1984)
	von zur Gathen & Gerhard (1999)
$R_k(f,g) = \sum_{0 \le i \le n} \det(M_{ik})x^i$	Collins (1967), Loos (1982)
	Geddes *et al.* (1992)
$= Z_k(f,g) = \det(M_k^*) \in R[x]$	Brown & Traub (1971)
	Zippel (1993), Lickteig & Roy (1997)
	Reischert (1997)

Table 1. Definitions of subresultants

(ii) $R_k(f,g) = Z_k(f,g)$.

Lemma 3.9. *Let F be a field, f and g in $F[x]$ be polynomials of degree $n \ge m > 0$, respectively, and let r_i, s_i and t_i be the entries in the ith row of the Extended Euclidean Scheme, so that $r_i = s_i f + t_i g$ for $0 \le i \le \ell$. Moreover, let $\rho_i = \operatorname{lc}(r_i)$ and $n_i = \deg r_i$ for all i. Then*

$$\frac{\sigma_{n_i}}{\rho_i} \cdot r_i = R_{n_i} \text{ for } 2 \le i \le \ell.$$

Remark 3.10. Let f and g be polynomials over an integral domain R, let F be the field of fractions of R, and consider the Extended Euclidean Scheme of f and g in $F[x]$. Then the scalar and the polynomial subresultants are in R and $R[x]$, respectively, and Lemma 3.9 also holds:

$$\frac{\sigma_{n_i}}{\rho_i} \cdot r_i = R_{n_i} \in R[x].$$

Note that r_i is not necessarily in $R[x]$, and ρ_i not necessarily in R.

4 Division rules and Polynomial Remainder Sequences (PRS)

We cannot directly apply the Euclidean Algorithm to polynomials f and g over an integral domain R since polynomial division with remainder in $R[x]$, which is used in every step of the Euclidean Algorithm, is not always defined. Hence our goal now are definitions modified in such a way that they yield a variant of the Euclidean Algorithm that works over an integral domain. We introduce a generalization of the usual pseudo-division, the concept of *division rules*, which leads to intermediate results in $R[x]$.

Definition 4.1. *Let R be an integral domain. A* one-step division rule *is a partial mapping*

$$\mathcal{R}\colon R[x]^2 \nrightarrow R^2$$

such that for all $(f,g) \in \operatorname{def}(\mathcal{R})$ there exist $q, r \in R[x]$ satisfying

(i) $\mathcal{R}(f,g) = (\alpha, \beta)$,
(ii) $\alpha f = qg + \beta r$ *and* $\deg r < \deg g$.

Recall that $\operatorname{def}(\mathcal{R}) \subseteq R[x]^2$ is the *domain of definition* of $\mathcal{R}$, that is, the set of $(f,g) \in R[x]^2$ at which $\mathcal{R}$ is defined. In particular, $\mathcal{R}\colon \operatorname{def}(\mathcal{R}) \longrightarrow R^2$ is a total map. In the examples below, we will usually define one-step division rules by starting with a (total or partial) map $\mathcal{R}_0\colon R[x]^2 \nrightarrow R^2$ and then taking $\mathcal{R}$ to be the maximal one-step division rule consistent with $\mathcal{R}_0$. Thus

$$\begin{aligned}\operatorname{def}(\mathcal{R}) = \{(f,g) \in R[x]^2 :& \exists \alpha, \beta \in R,\ \exists q, r \in R[x] \\ & (\alpha,\beta) = \mathcal{R}_0(f,g) \text{ and (ii) holds}\},\end{aligned}$$

and $\mathcal{R}$ is $\mathcal{R}_0$ restricted to $\operatorname{def}(\mathcal{R})$. Furthermore $(f,0)$ is never in $\operatorname{def}(\mathcal{R})$ ("you can't divide by zero"), so that

$$\operatorname{def}(\mathcal{R}) \subseteq \mathcal{D}_{\max} = R[x] \times (R[x] \setminus \{0\}).$$

We are particularly interested in one-step division rules $\mathcal{R}$ with $\operatorname{def}(\mathcal{R}) = \mathcal{D}_{\max}$. In our examples, $(0,g)$ will always be in $\operatorname{def}(\mathcal{R})$ if $g \neq 0$.

We may consider the usual remainder as a partial function rem: $R[x]^2 \nrightarrow R[x]$ with $\operatorname{rem}(f,g) = r$ if there exist $q, r \in R[x]$ with $f = qg + r$ and $\deg r < \deg g$, and def(rem) maximal. Recall from Section 2 the definitions of rem, prem and cont.

Example 4.2. Let f and g be polynomials over an integral domain R of degrees n and m, respectively, and let $f_n = \operatorname{lc}(f)$, $g_m = \operatorname{lc}(g) \neq 0$ be their leading coefficients. Then the three most famous types of division rules are as follows:

- *classical division rule*: $\mathcal{R}(f,g) = (1,1)$.
- *monic division rule*: $\mathcal{R}(f,g) = (1, \operatorname{lc}(\operatorname{rem}(f,g)))$.
- *Sturmian division rule*: $\mathcal{R}(f,g) = (1,-1)$.

Examples are given below. When R is a field, these three division rules have the largest possible domain of definition $\operatorname{def}(\mathcal{R}) = \mathcal{D}_{\max}$, but otherwise, it may be smaller; we will illustrate this in Example 4.7. Hence they do not help us in achieving our goal of finding rules with maximal domain $\mathcal{D}_{\max}$. But there exist two division rules which, in contrast to the first examples, always yield solutions in $R[x]$:

- *pseudo-division rule*: $\mathcal{R}(f,g) = (g_m^{n-m+1}, 1)$.

In case R is a unique factorization domain, we have the

- *primitive division rule*: $\mathcal{R}(f,g) = (g_m^{n-m+1}, \operatorname{cont}(\operatorname{prem}(f,g)))$.

For algorithmic purposes, it is then useful for R to be a Euclidean domain.

The disadvantage of the pseudo-division rule, however, is that in the Euclidean Algorithm it leads to exponential coefficient growth; the coefficients of the intermediate results are usually enormous, their bit length may be exponential in the bit length of the input polynomials f and g. If R is a UFD, we get the smallest intermediate results if we use the primitive division rule, but the computation of the content in every step of the Euclidean Algorithm seems to be expensive. Collins (1967) already observed this in his experiments. Thus he tries to avoid the computation of contents and to keep the intermediate results "small" at the same time by using information from *all* intermediate results in the EEA, not only the two previous remainders. Our concept of one-step division rules does not cover his method. So we now extend our previous definition, and will actually capture all the "recursive" division rules from Collins (1967, 1971, 1973), Brown & Traub (1971) and Brown (1971) under one umbrella.

Definition 4.3. *Let R be an integral domain. A* division rule *is a partial mapping*

$$\mathcal{R}\colon R[x]^2 \rightharpoonup (R^2)^*$$

associating to $(f,g) \in \operatorname{def}(\mathcal{R})$ a sequence $((\alpha_2,\beta_2),\ldots,(\alpha_{\ell+1},\beta_{\ell+1}))$ of arbitrary length ℓ such that for all $(f,g) \in \operatorname{def}(\mathcal{R})$ there exist $\ell \in \mathbb{N}_{\geq 0}$, $q_1,\ldots,q_\ell \in R[x]$ and $r_0,\ldots,r_{\ell+1} \in R[x]$ satisfying

(i) $r_0 = f, r_1 = g$,
(ii) $\mathcal{R}_i(f,g) = \mathcal{R}(f,g)_i = (\alpha_i,\beta_i)$,
(iii) $\alpha_i r_{i-2} = q_{i-1}r_{i-1} + \beta_i r_i$ *and* $\deg r_i < \deg r_{i-1}$

for $2 \leq i \leq \ell+1$. The integer $\ell = |\mathcal{R}(f,g)|$ is the length *of the sequence.*

A division rule where $\ell = 1$ for all values is the same as a one-step division rule, and from an arbitrary division rule we can obtain a one-step division rule by projecting to the first coordinate (α_2,β_2) if $\ell \geq 2$. Using Lemma 2.6, we find that for all $(f,g) \in \operatorname{def}(\mathcal{R})$, q_{i-1} and r_i are unique for $2 \leq i \leq \ell+1$. If we have a one-step division rule $\mathcal{R}^*$ which is defined at all (r_{i-2},r_{i-1}) for $2 \leq i \leq \ell+1$ (defined recursively), then we obtain a division rule $\mathcal{R}$ by using $\mathcal{R}^*$ in every step:

$$\mathcal{R}_i(f,g) = \mathcal{R}^*(r_{i-2},r_{i-1}) = (\alpha,\beta).$$

If we truncate $\mathcal{R}$ at the first coordinate, we get $\mathcal{R}^*$ back. But the notion of division rules is strictly richer than that of one-step division rules; for example the first step in the reduced division rule below is just the pseudo-division rule, but using the pseudo-division rule repeatedly does not yield the reduced division rule.

Example 4.2 continued. Let $f = r_0, g = r_1, r_2, \ldots, r_\ell \in R[x]$ be as in Definition 4.3, let $n_i = \deg r_i$ be their degrees, $\rho_i = \operatorname{lc}(r_i)$ their leading coefficients, and $d_i = n_i - n_{i+1} \in \mathbb{N}_{\geq 0}$ for $0 \leq i \leq \ell$ (if $n_0 \geq n_1$). We now present two different types of recursive division rules. They are based on polynomial subresultants. It is not obvious that they have domain of definition $\mathcal{D}_{\max}$, since divisions occur in their definitions. We will show that this is indeed the case in Remarks 6.8 and 6.12.

- *reduced division rule*: $\mathcal{R}_i(f,g) = (\alpha_i, \beta_i)$ for $2 \le i \le \ell+1$, where we set $\alpha_1 = 1$ and

$$(\alpha_i, \beta_i) = (\rho_{i-1}^{d_{i-2}+1}, \alpha_{i-1}) \text{ for } 2 \le i \le \ell+1.$$

- *subresultant division rule*: $\mathcal{R}_i(f,g) = (\alpha_i, \beta_i)$ for $2 \le i \le \ell+1$, where we set $\rho_0 = 1$, $\psi_2 = -1$, $\psi_3, \ldots, \psi_{\ell+1} \in R$ with

$$\begin{aligned}(\alpha_i, \beta_i) &= (\rho_{i-1}^{d_{i-2}+1}, -\rho_{i-2}\psi_i^{d_{i-2}}) \text{ for } 2 \le i \le \ell+1,\\ \psi_i &= (-\rho_{i-2})^{d_{i-3}}\psi_{i-1}^{1-d_{i-3}} \text{ for } 3 \le i \le \ell+1.\end{aligned}$$

The subresultant division rule was invented by Collins (1967), p. 130. He tried to find a rule such that the r_i's agree with the polynomial subresultants up to a small constant factor. Brown (1971), p. 486, then provided a recursive definition of the α_i and β_i as given above. Brown (1971) also describes an "improved division rule", where one has some magical divisor of ρ_i.

We note that the exponents in the recursive definition of the ψ_i's in the subresultant division rule may be negative. Hence it is not clear that the β_i's are in R. However, we will show this in Theorem 6.15, and so answer the following open question that was posed in Brown (1971), p. 486:

Question 4.4. *"At the present time it is not known whether or not these equations imply $\psi_i, \beta_i \in R$."*

By definition, a division rule $\mathcal{R}$ defines a sequence $(r_0, \ldots, r_\ell)$ of remainders; recall that they are uniquely defined. Since it is more convenient to work with these "polynomial remainder sequences", we fix this notion in the following definition.

Definition 4.5. *Let $\mathcal{R}$ be a division rule. A sequence $(r_0, \ldots, r_\ell)$ with each $r_i \in R[x] \setminus \{0\}$ is called a* polynomial remainder sequence (PRS) for (f,g) according to $\mathcal{R}$ *if*

(i) $r_0 = f, r_1 = g$,
(ii) $\mathcal{R}_i(f,g) = (\alpha_i, \beta_i)$,
(iii) $\alpha_i r_{i-2} = q_{i-1} r_{i-1} + \beta_i r_i$,

for $2 \le i \le \ell+1$, where ℓ is the length *of $\mathcal{R}(f,g)$. The PRS is* complete *if $r_{\ell+1} = 0$. It is called* normal *if $d_i = \deg r_i - \deg r_{i+1} = 1$ for $1 \le i \le \ell-1$ (Collins (1967), p. 128/129).*

In fact the remainders for PRS according to arbitrary division rules over an integral domain only differ by a nonzero constant factor.

Proposition 4.6. *Let R be an integral domain, $f, g \in R[x]$ and $r = (r_0, \ldots, r_\ell)$ and $r^* = (r_0^*, \ldots, r_{\ell^*}^*)$ be two PRS for (f, g) according to two division rules $\mathcal{R}$ and $\mathcal{R}^*$, respectively, none of whose results $\alpha_i, \beta_i, \alpha_i^*, \beta_i^*$ is zero. Then $r_i^* = \gamma_i r_i$ with*

$$\gamma_i = \prod_{0 \le k \le i/2-1} \frac{\alpha_{i-2k}^* \beta_{i-2k}}{\alpha_{i-2k} \beta_{i-2k}^*} \in F \setminus \{0\}$$

for $0 \le i \le \min\{\ell, \ell^*\}$, where F is the field of fractions of R.

The proposition yields a direct way to compute the PRS for (f, g) according to $\mathcal{R}^*$ from the PRS for (f, g) according to $\mathcal{R}$ and the $\alpha_i, \beta_i, \alpha_i^*, \beta_i^*$. In particular, the degrees of the remainders in any two PRS are identical.

In Example 4.2 we have seen seven different division rules. Now we consider the different polynomial remainder sequences according to these rules. Each PRS will be illustrated by the following example.

Example 4.7. We perform the computations on the polynomials

$$\begin{aligned} f &= r_0 = 9x^6 - 27x^4 - 27x^3 + 72x^2 + 18x - 45 \text{ and} \\ g &= r_1 = 3x^4 - 4x^2 - 9x + 21 \end{aligned}$$

over $R = \mathbb{Q}$ and, wherever possible, also over $R = \mathbb{Z}$.

i	classical	monic	Sturmian	pseudo
0	$9x^6 - 27x^4 - 27x^3 + 72x^2 + 18x - 45$			
1	$3x^4 - 4x^2 - 9x + 21$			
2	$-11x^2 - 27x + 60$	$x^2 + \frac{27}{11}x - \frac{60}{11}$	$11x^2 - 27x + 60$	$-297x^2 - 729x + 1620$
3	$-\frac{164\,880}{1331}x + \frac{248\,931}{1331}$	$x - \frac{27\,659}{18\,320}$	$\frac{164\,880}{1331}x + \frac{248\,931}{1331}$	$3\,245\,333\,040x - 4\,899\,708\,873$
4	$-\frac{1\,959\,126\,851}{335\,622\,400}$	1	$-\frac{1\,959\,126\,851}{335\,622\,400}$	$-1\,659\,945\,865\,306\,233\,453\,993$

i	primitive	reduced	subresultant
0	$9x^6 - 27x^4 - 27x^3 + 72x^2 + 18x - 45$		
1	$3x^4 - 4x^2 - 9x + 21$		
2	$-11x^2 - 27x + 60$	$-297x^2 - 729x + 1620$	$297x^2 + 729x - 1620$
3	$18\,320x - 27\,659$	$120\,197\,520x - 181\,470\,699$	$13\,355\,280x - 20\,163\,411$
4	-1	$86\,915\,463\,129$	$9\,657\,273\,681$

1. **Classical PRS.** The most familiar PRS for (f, g) is obtained according to the *classical division rule*. Collins (1973), p. 736, calls this the *natural Euclidean PRS (algorithm)*. The intermediate results of the classical PRS and of the Euclidean Algorithm coincide.
2. **Monic PRS.** In Collins (1973), p. 736, the PRS for (f, g) according to the *monic division rule* is called *monic PRS (algorithm)*. The r_i are monic for $2 \le i \le \ell$, and we get the same intermediate results as in the *monic Euclidean Algorithm* in von zur Gathen & Gerhard (1999), Section 3.2, p. 47.

3. **Sturmian PRS.** We choose the PRS for (f, g) according to the *Sturmian division rule* as introduced in Sturm (1835). Kronecker (1873), p. 117, Habicht (1948), p. 102, and Loos (1982), p. 119, deal with this *generalized Sturmian PRS (algorithm)*. Kronecker (1873) calls it *Sturmsche Reihe (Sturmian sequence)*, and in Habicht (1948) it is the *verallgemeinerte Sturmsche Kette (generalized Sturmian chain)*. If $g = \partial f/\partial x$ as in Habicht (1948), p. 99, then this is the *classical Sturmian PRS (algorithm)*. Note that the Sturmian PRS agrees with the classical PRS up to sign.

If R is not a field, then Example 4.7 shows that the first three types of PRS may not have $\mathcal{D}_{\max}$ as their domain of definition. In the example they are only of length 1. But fortunately there are division rules that have this property.

4. **Pseudo PRS.** If we choose the PRS according to the *pseudo-division rule*, then we get the so-called *pseudo PRS*. Collins (1967), p. 138, calls this the *Euclidean PRS (algorithm)* because it is the most obvious generalization of the Euclidean Algorithm to polynomials over an integral domain R that is not a field. Collins (1973), p. 737, also calls it *pseudo-remainder PRS*.
5. **Primitive PRS.** To obtain a PRS over R with minimal coefficient growth, we choose the PRS according to the *primitive division rule* which yields primitive intermediate results. Brown (1971), p. 484, calls this the *primitive PRS (algorithm)*.
6. **Reduced PRS.** A perceived drawback of the primitive PRS is the (seemingly) costly computation of the content; recently the algorithm of Cooperman *et al.* (1999) achieves this with an expected number of less than two integer gcd's. In fact, in our experiments in Section 8, the primitive PRS turns out to be most efficient among those discussed here. But Collins (1967) introduced his *reduced PRS (algorithm)* in order to avoid the computation of contents completely. His algorithm uses the *reduced division rule* and keeps the intermediate coefficients reasonably small but not necessarily as small as with the primitive PRS.
7. **Subresultant PRS.** The reduced PRS is not the only way to keep the coefficients small without computing contents. We can also use the *subresultant division rule*. According to Collins (1967), p. 130, this is the *subresultant PRS (algorithm)*.

5 Fundamental Theorem on subresultants

Collins' Fundamental Theorem on subresultants expresses an arbitrary subresultant as a power product of certain data in the PRS, namely the multipliers α and β and the leading coefficients of the remainders in the Euclidean Algorithm. In this section our first goal is to prove the Fundamental Theorem on subresultants for polynomial remainder sequences according to an arbitrary division rule $\mathcal{R}$.

The following result is shown for PRS in Brown & Traub (1971), p. 511, Fundamental theorem, and for reduced PRS in Collins (1967), p. 132, Lemma 2, and p. 133, Theorem 1.

Fundamental Theorem 5.1. *Let f and $g \in R[x]$ be polynomials of degrees $n \geq m > 0$, respectively, over an integral domain R, let $\mathcal{R}$ be a division rule and $(r_0, \ldots, r_\ell)$ be the PRS for (f, g) according to $\mathcal{R}$, $(\alpha_i, \beta_i) = \mathcal{R}_i(f, g)$ the constant multipliers, $n_i = \deg r_i$ and $\rho_i = \mathrm{lc}(r_i)$ for $0 \leq i \leq \ell$, and $d_i = n_i - n_{i+1}$ for $0 \leq i \leq \ell - 1$.*

(i) For $0 \leq j \leq n_1$, the jth subresultant of (f, g) is

$$\sigma_j(f,g) = (-1)^{b_i} \rho_i^{n_{i-1}-n_i} \prod_{2 \leq k \leq i} \left(\frac{\beta_k}{\alpha_k}\right)^{n_{k-1}-n_i} \rho_{k-1}^{n_{k-2}-n_k}$$

if $j = n_i$ for some $1 \leq i \leq \ell$, otherwise 0, where $b_i = \sum_{2 \leq k \leq i} (n_{k-2} - n_i)(n_{k-1} - n_i)$.

(ii) The subresultants satisfy for $1 \leq i < \ell$ the recursive formulas

$$\sigma_{n_1}(f,g) = \rho_1^{d_0} \text{ and}$$
$$\sigma_{n_{i+1}}(f,g) = \sigma_{n_i}(f,g) \cdot (-1)^{d_i(n_0 - n_{i+1} + i + 1)} (\rho_{i+1}\rho_i)^{d_i} \prod_{2 \leq k \leq i+1} \left(\frac{\beta_k}{\alpha_k}\right)^{d_i}.$$

Corollary 5.2. *Let $\mathcal{R}$ be a division rule and $(r_0, \ldots, r_\ell)$ be the PRS for (f, g) according to $\mathcal{R}$, let $n_i = \deg r_i$ for $0 \leq i \leq \ell$ be the degrees in the PRS, and let $0 \leq k \leq n_1$. Then*

$$\sigma_k \neq 0 \iff \exists i \colon k = n_i.$$

6 Applications of the Fundamental Theorem

Following our program, we now derive results for the various PRS for polynomials $f, g \in R[x]$ of degrees $n \geq m \geq 0$, respectively, over an integral domain R, according to the division rules in Example 4.2.

Corollary 6.1. *Let $(r_0, \ldots, r_\ell)$ be a* **classical PRS** *and $1 \leq i \leq \ell$. Then*

(i)
$$\sigma_{n_i}(f,g) = (-1)^{b_i} \rho_i^{d_{i-1}} \prod_{2 \leq k \leq i} \rho_{k-1}^{n_{k-2}-n_k}.$$

(ii) The subresultants satisfy the recursive formulas

$$\sigma_{n_1}(f,g) = \rho_1^{d_0}, \text{ and}$$
$$\sigma_{n_{i+1}}(f,g) = \sigma_{n_i}(f,g) \cdot (-1)^{d_i(n_0 - n_{i+1} + i + 1)} (\rho_{i+1}\rho_i)^{d_i}.$$

If the PRS is normal, then this simplifies to:

(iii)
$$\sigma_{n_i}(f,g) = (-1)^{(d_0+1)(i+1)} \rho_i \rho_1^{d_0+1} \prod_{3 \leq k \leq i} \rho_{k-1}^2 \text{ for } i \geq 2.$$

(iv) The subresultants satisfy the recursive formulas

$$\sigma_{n_1}(f,g) = \rho_1^{d_0}, \text{ and}$$
$$\sigma_{n_{i+1}}(f,g) = \sigma_{n_i}(f,g) \cdot (-1)^{d_0+1} \rho_{i+1}\rho_i.$$

The following is the Fundamental Theorem 11.13 in von zur Gathen & Gerhard (1999), Chapter 11.2, p. 307.

Corollary 6.2. *Let $(r_0, \ldots, r_\ell)$ be a* **monic PRS**, *and $1 \le i \le \ell$. Then*

(i)
$$\sigma_{n_i}(f,g) = (-1)^{b_i} \rho_0^{n_1-n_i} \rho_1^{n_0-n_i} \prod_{2 \le k \le i} \beta_k^{n_{k-1}-n_i}.$$

(ii) The subresultants satisfy the recursive formulas

$$\begin{aligned}&\sigma_{n_1}(f,g) = \rho_1^{d_0}, \text{ and}\\ &\sigma_{n_{i+1}}(f,g) = \sigma_{n_i}(f,g) \cdot (-1)^{d_i(n_0-n_{i+1}+i+1)} (\rho_0\rho_1\beta_2 \cdots \beta_{i+1})^{d_i}.\end{aligned}$$

If the PRS is normal, then this simplifies to:

(iii)
$$\sigma_{n_i}(f,g) = (-1)^{(d_0+1)(i+1)} \rho_0^{i-1} \rho_1^{d_0+i-1} \prod_{2 \le k \le i} \beta_k^{i-(k-1)} \text{ for } i \ge 2.$$

(iv) The subresultants satisfy for $1 \le i < \ell$ the recursive formulas

$$\begin{aligned}&\sigma_{n_1}(f,g) = \rho_1^{d_0}, \text{ and}\\ &\sigma_{n_{i+1}}(f,g) = \sigma_{n_i}(f,g) \cdot (-1)^{d_0+1} \rho_0\rho_1\beta_2 \cdots \beta_{i+1}.\end{aligned}$$

Corollary 6.3. *Let $(r_0, \ldots, r_\ell)$ be a* **Sturmian PRS**, *and $1 \le i \le \ell$. Then*

(i)
$$\sigma_{n_i}(f,g) = (-1)^{b_i + \sum_{2 \le k \le i}(n_{k-1}-n_i)} \rho_i^{d_{i-1}} \prod_{2 \le k \le i} \rho_{k-1}^{n_{k-2}-n_k}.$$

(ii) The subresultants satisfy the recursive formulas

$$\begin{aligned}&\sigma_{n_1}(f,g) = \rho_1^{d_0}, \text{ and}\\ &\sigma_{n_{i+1}}(f,g) = \sigma_{n_i}(f,g) \cdot (-1)^{d_i(n_0-n_{i+1}+1)} (\rho_{i+1}\rho_i)^{d_i}.\end{aligned}$$

If the PRS is normal, then this simplifies to:

(iii)
$$\sigma_{n_i}(f,g) = (-1)^{(d_0+1)(i+1)} \rho_1^{d_0+1} \rho_i \prod_{3 \le k \le i} \rho_{k-1}^2 \text{ for } i \ge 2.$$

(iv) The subresultants satisfy the recursive formulas

$$\begin{aligned}&\sigma_{n_1}(f,g) = \rho_1^{d_0}, \text{ and}\\ &\sigma_{n_{i+1}}(f,g) = \sigma_{n_i}(f,g) \cdot (-1)^{d_0+i+1} \rho_{i+1}\rho_i.\end{aligned}$$

The following corollary can be found in Collins (1966), p. 710, Theorem 1, for polynomial subresultants.

Corollary 6.4. *Let $(r_0, \ldots, r_\ell)$ be a* **pseudo PRS**, *and $1 \le i \le \ell$. Then*

(i)
$$\sigma_{n_i}(f,g) = (-1)^{b_i} \rho_i^{d_{i-1}} \prod_{2 \le k \le i} \rho_{k-1}^{n_{k-2}-n_k-(n_{k-1}-n_i)(d_{k-2}+1)}.$$

(ii) The subresultants satisfy the recursive formulas

$$\sigma_{n_1}(f,g) = \rho_1^{d_0}, \text{ and}$$
$$\sigma_{n_{i+1}}(f,g) = \sigma_{n_i}(f,g)\cdot(-1)^{d_i(n_0-n_{i+1}+i+1)}(\rho_{i+1}\rho_i)^{d_i}\prod_{2\le k\le i+1}\rho_{k-1}^{-(d_{k-2}+1)d_i}.$$

If the PRS is normal, then this simplifies to:

(iii) $$\sigma_{n_i}(f,g) = (-1)^{(d_0+1)(i+1)}\rho_1^{(d_0+1)(2-i)}\rho_i\prod_{3\le k\le i-1}\rho_{k-1}^{2(k-i)} \text{ for } i\ge 2.$$

(iv) The subresultants satisfy the recursive formulas

$$\sigma_{n_1}(f,g) = \rho_1^{d_0}, \text{ and}$$
$$\sigma_{n_{i+1}}(f,g) = \sigma_{n_i}(f,g)\cdot(-1)^{d_0+1}\rho_1^{-(d_0+1)}\rho_{i+1}\rho_i\prod_{3\le k\le i+1}\rho_{k-1}^{-2}.$$

Remark 6.5. If the PRS is normal, then Corollary 6.4(iii) implies that

$$\sigma_{n_i}(f,g)(-1)^{(\delta_0+1)(i+1)}\rho_1^{(d_0+1)(i-2)}\prod_{3\le k\le i-1}\rho_{k-1}^{2(i-k)} = \rho_i.$$

Thus $\sigma_{n_i}(f,g)$ divides ρ_i. This result is also shown for polynomial subresultants in Collins (1966), p. 711, Corollary 1.

Since the content of two polynomials cannot be expressed in terms of our parameters ρ_i and n_i, we do not consider the Fundamental theorem for **primitive PRS**.

The following is shown for polynomial subresultants in Collins (1967), p. 135, Corollaries 1.2 and 1.4.

Corollary 6.6. *Let $(r_0,\dots,r_\ell)$ be a* **reduced PRS**, *and $1\le i\le \ell$. Then*

(i) $$\sigma_{n_i}(f,g) = (-1)^{b_i}\rho_i^{d_{i-1}}\prod_{2\le k\le i}\rho_{k-1}^{d_{k-2}(1-d_{k-1})}$$

(ii) The subresultants satisfy for the recursive formulas

$$\sigma_{n_1}(f,g) = \rho_1^{d_0}, \text{ and}$$
$$\sigma_{n_{i+1}}(f,g) = \sigma_{n_i}(f,g)\cdot(-1)^{d_i(n_0-n_{i+1}+i+1)}\rho_{i+1}^{d_i}\rho_i^{-d_{i-1}d_i}.$$

If the PRS is normal, then this simplifies to:

(iii) $$\sigma_{n_i}(f,g) = (-1)^{(d_0+1)(i+1)}\rho_i \text{ for } i\ge 2.$$

(iv) The subresultants satisfy the recursive formulas

$$\sigma_{n_1}(f,g) = \rho_1^{d_0}, \text{ and}$$
$$\sigma_{n_{i+1}}(f,g) = \sigma_{n_i}(f,g)\cdot(-1)^{d_0+1}\rho_{i+1}\rho_i^{-1}.$$

Remark 6.7. We obtain from Corollary 6.6(i)

$$\sigma_{n_i}(f,g) \prod_{2\leq k\leq i} (-1)^{(n_{k-2}-n_i)(n_{k-1}-n_i)} \rho_{k-1}^{d_{k-2}(d_{k-1}-1)} = \rho_i^{d_{i-1}}.$$

Thus $\sigma_{n_i}(f,g)$ divides $\rho_i^{d_{i-1}}$. This result can also be found in Collins (1967), p.135, Corollary 1.2.

Remark 6.8. For every reduced PRS, r_i is in $R[x]$ for $2 \leq i \leq \ell$. Note that Corollary 6.6(iii) implies $r_i = (-1)^{(d_0+1)(i+1)} R_i(f,g)$. So the normal case is clear. An easy proof for the general case based on polynomial subresultants is in Collins (1967), p. 134, Corollary 1.1, and Brown (1971), p. 485/486.

Lemma 6.9. *Let $e_{i,j} = d_{j-1} \prod_{j\leq k\leq i}(1-d_k)$, and let ψ_i be as in the subresultant division rule. Then*

$$\psi_i = -\prod_{1\leq j\leq i-2} \rho_j^{e_{i-3,j}} \text{ for } 2\leq i\leq \ell.$$

Corollary 6.10. *Let $(r_0,\dots,r_\ell)$ be a* **subresultant PRS**, *and $1\leq i\leq \ell$. Then*

(i)
$$\sigma_{n_i}(f,g) = \prod_{1\leq k\leq i} \rho_k^{e_{i-1,k}}.$$

(ii) The subresultants satisfy the recursive formulas

$$\sigma_{n_1}(f,g) = \rho_1^{d_0}, \text{ and}$$
$$\sigma_{n_{i+1}}(f,g) = \sigma_{n_i}(f,g)\cdot \rho_{i+1}^{d_i} \prod_{1\leq k\leq i} \rho_k^{-d_i e_{i-1,k}}.$$

If the PRS is normal, then this simplifies to:

(iii)
$$\sigma_{n_i}(f,g) = \rho_i \text{ for } i\geq 2.$$

(iv) The subresultants satisfy the recursive formulas

$$\sigma_{n_1}(f,g) = \rho_1^{d_0}, \text{ and}$$
$$\sigma_{n_{i+1}}(f,g) = \sigma_{n_i}(f,g)\cdot \rho_{i+1}\rho_i^{-1}.$$

Now we have all tools to prove the relation between normal reduced and normal subresultant PRS which can be found in Collins (1967), p. 135, Corollary 1.3, and Collins (1973), p. 738.

Corollary 6.11. *Let $(r_0,\dots,r_\ell)$ be a normal reduced PRS and $(a_0,\dots,a_\ell)$ a normal subresultant PRS for the polynomials $r_0 = a_0 = f$ and $r_1 = a_1 = g$. Then the following holds for $2\leq i\leq \ell$:*

$$\mathrm{lc}(r_i) = (-1)^{(n_0-n_i)(n_1-n_i)}\cdot \mathrm{lc}(a_i).$$

Remark 6.12. For every subresultant PRS the polynomials r_i are in $R[x]$ for $2 \leq i \leq \ell$. Note that Corollary 6.10(iii) implies $r_i = R_i(f, g)$. So the normal case is clear. An easy proof for the general case based on polynomial subresultants is in Collins (1967), p. 130, and Brown (1971), p. 486.

Corollary 6.10 does not provide the only recursive formula for subresultants. Another one is based on an idea in Lickteig & Roy (1997), p. 12, and Reischert (1997), p. 238, where the following formula has been proven for polynomial subresultants. The translation of this result into a theorem on scalar subresultants leads us to an answer to Question 4.4.

Theorem 6.13. *Let $(r_0, \dots, r_\ell)$ be a subresultant PRS. Then the subresultants satisfy for $1 \leq i < \ell$ the recursive formulas*

$$\begin{aligned}&\sigma_{n_1}(f, g) = \rho_1^{d_0} \text{ and}\\ &\sigma_{n_{i+1}}(f, g) = \sigma_{n_i}(f, g)^{1-d_i} \cdot \rho_{i+1}^{d_i}.\end{aligned}$$

The proof of the conjecture now becomes pretty simple:

Corollary 6.14. *Let $\psi_2 = -1$ and $\psi_i = (-\rho_{i-2})^{d_{i-3}} \psi_{i-1}^{1-d_{i-3}}$ for $3 \leq i \leq \ell$. Then*

$$\psi_i = -\sigma_{n_{i-2}}(f, g) \text{ for } 3 \leq i \leq \ell.$$

Since all subresultants are in R, this gives an answer to Question 4.4:

Theorem 6.15. *The coefficients ψ_i and β_i of the subresultant PRS are always in R.*

7 Analysis of coefficient growth and running time

We first estimate the running times for normal PRS. A proof for an exponential *upper* bound for the pseudo PRS is in Knuth (1981), p. 414, but our goal is to show an exponential *lower* bound. To this end, we prove two such bounds on the bit length of the leading coefficients ρ_i in this PRS. Recall that $\rho_1 = \mathrm{lc}(g)$ and $\sigma_{n_1} = \rho_1^{\delta_0+1}$.

Lemma 7.1. *Suppose that $(f, g) \in \mathbb{Z}[x]^2$ have a normal pseudo PRS. Then*

$$|\rho_i| \geq |\rho_1|^{2^{i-3}} \text{ for } 3 \leq i \leq \ell.$$

Lemma 7.2. *Suppose that $(f, g) \in \mathbb{Z}[x]^2$ have a normal pseudo PRS, and that $|\rho_1| = 1$. Then*

$$|\rho_i| \geq |\sigma_{n_i} \prod_{2 \leq k \leq i-2} \sigma_{n_k}(f, g)^{2^{i-k-1}}| \text{ for } 3 \leq i \leq \ell.$$

Theorem 7.3. *Computing the pseudo PRS takes exponential time, at least 2^n, in some cases with input polynomials of degrees at most n.*

We have the following running time bound for the normal reduced PRS algorithm.

Theorem 7.4. *Let $\|f\|_\infty$, $\|g\|_\infty \leq A$, $B = (n+1)^n A^{n+m}$, and let $(r_0, \ldots, r_\ell)$ be the normal reduced PRS for f, g. Then the max-norm of the r_i is at most $4B^3$, and the algorithm uses $O(n^3 m \log^2(nA))$ word operations.*

Corollary 7.5. *Since Corollary 6.11 shows that normal reduced PRS and normal subresultant PRS agree up to sign, the estimates in Theorem 7.4 are also true for normal subresultant PRS.*

We conclude the theoretical part of our comparison with an overview of all worst-case running times for the various normal PRS in Table 2. The length of the coefficients of f and g are assumed to be at most n. The estimations that are not proven here can be found in von zur Gathen & Gerhard (1999).

PRS	**time**	**proven in**
classical/Sturmian	n^8	von zur Gathen & Gerhard (1999)
monic	n^6	von zur Gathen & Gerhard (1999)
pseudo	c^n with $c \geq 2$	Theorem 7.3
primitive	n^6	von zur Gathen & Gerhard (1999)
reduced/subresultant	n^6	Theorem 7.4

Table 2. Comparison of various normal PRS. The time in bit operations is for polynomials of degree at most n and with coefficients of length at most n and ignores logarithmic factors.

8 Experiments

We have implemented six of the PRS for polynomials with integral coefficients in C++, using Victor Shoup's "Number Theory Library" NTL 3.5a for integer and polynomial arithmetic. Since the Sturmian PRS agrees with the classical PRS up to sign, it is not mentioned here. The contents of the intermediate results in the primitive PRS are simply computed by successive gcd computations. Cooperman *et al.* (1999) propose a new algorithm that uses only an expected number of two gcd computations, but on random inputs it is slower than the naïve approach. All timings are the average over 10 pseudorandom inputs. The software ran on a Sun Sparc Ultra 1 clocked at 167MHz.

In the first experiment we pseudorandomly and independently chose three polynomials $f, g, h \in \mathbb{Z}[x]$ of degree $n-1$ with nonnegative coefficients of length less than n, for various values of n. Then we used the various PRS algorithms

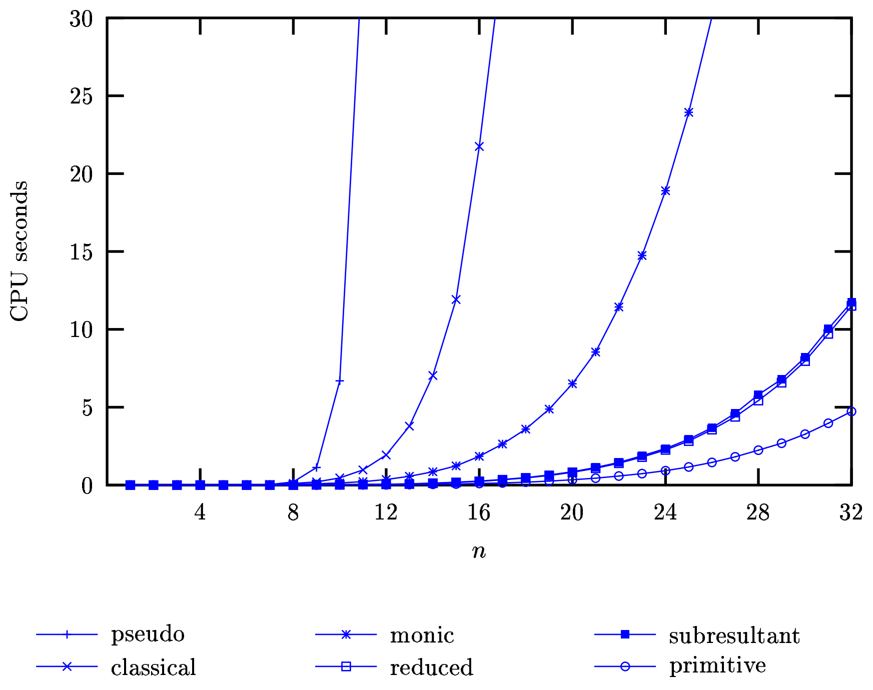

Fig. 1. Computation of polynomial remainder sequences for polynomials of degree $n-1$ with coefficients of bit length less than n for $1 \leq n \leq 32$.

to compute the gcd of fh and gh of degrees less than $2n$. The running times are shown in Figures 1 and 2.

As seen in Table 2 the pseudo PRS turns out to be the slowest algorithm. The reason is that for random inputs with coefficients of length at most n the second polynomial is almost never monic. Thus Theorem 7.3 shows that for random inputs the running time for pseudo PRS is mainly exponential. A surprising result is that the primitive PRS, even implemented in a straightforward manner, turns out to be the fastest PRS. Collins and Brown & Traub only invented the subresultant PRS in order to avoid the primitive PRS since it seemed too expensive, but our tests show that for our current software this is not a problem.

Polynomial remainder sequences of random polynomials tend to be normal. Since Corollary 6.11 shows that reduced and subresultant PRS agree up to signs in the normal case, their running times also differ by little.

We are also interested in comparing the reduced and subresultant PRS, so we construct PRS which are not normal. To this end, we pseudorandomly and independently choose six polynomials f, f_1, g, g_1, h, h_1 for various degrees n as follows:

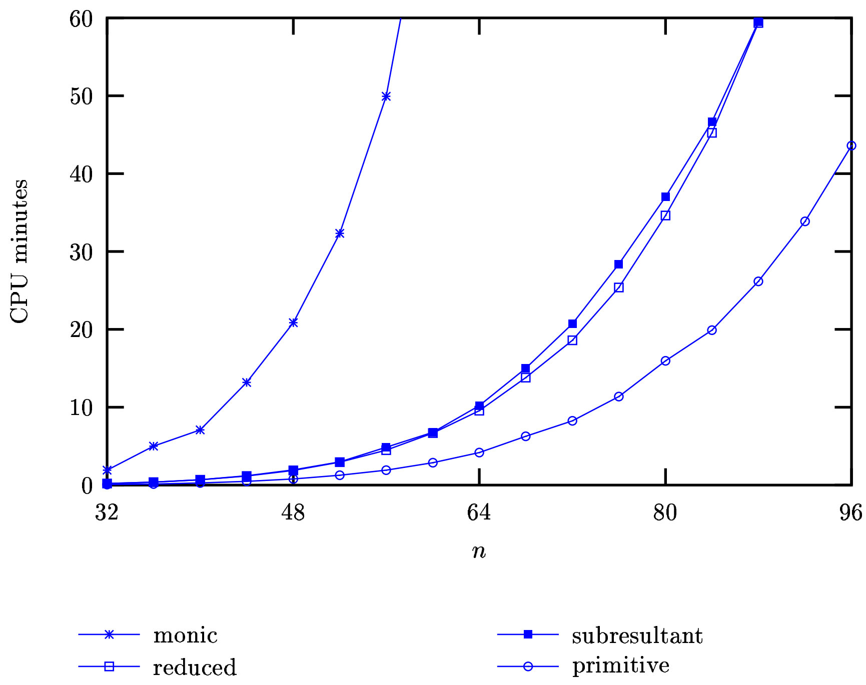

Fig. 2. Computation of polynomial remainder sequences for polynomials of degree $n-1$ with coefficients of bit length less than n for $32 \le n \le 96$. Time is now measured in minutes.

polynomial	degree	coefficient length
f, g	$n/6$	$n/4$
f_1, g_1	$n/3$	n
h	$n/2$	$3n/4$
h_1	n	n

So the polynomials

$$F = (fh \cdot x^n + f_1)h_1$$
$$G = (gh \cdot x^n + g_1)h_1$$

have degrees less than $2n$ with coefficient length less than n, and every polynomial remainder sequence of F and G has a degree jump of $\frac{n}{3}$ at degree $2n - \frac{n}{6}$. Then we used the various PRS algorithms to compute the gcd of F and G. The running times are illustrated in Figures 3 and 4.

As in the first test series the pseudo PRS turns out to be the slowest, and the primitive PRS is the fastest. Here the monic PRS is faster than the reduced PRS. Since the PRS is non-normal, the α_i's are powers of the leading coefficients of the intermediate results, and their computation becomes quite expensive.

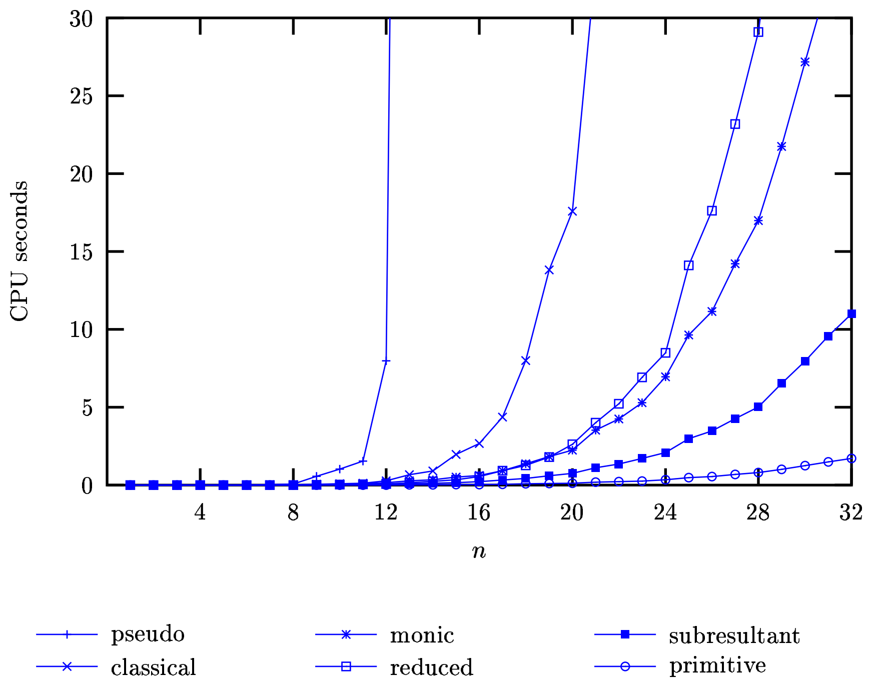

Fig. 3. Computation of non-normal polynomial remainder sequences for polynomials of degree $2n - 1$ with coefficient length less than n and a degree jump of $\frac{n}{3}$ at degree $2n - \frac{n}{6}$, for $1 \le n \le 32$.

References

Étienne Bézout, Recherches sur le degré des équations résultantes de l'évanouissement des inconnues. *Histoire de l'académie royale des sciences* (1764), 288–338. Summary 88–91.

Otto Biermann, Über die Resultante ganzer Functionen. *Monatshefte fuer Mathematik und Physik* (1891), 143–146. II. Jahrgang.

W. S. Brown, On Euclid's Algorithm and the Computation of Polynomial Greatest Common Divisors. *Journal of the ACM* **18**(4) (1971), 478–504.

W. S. Brown, The Subresultant PRS Algorithm. *ACM Transactions on Mathematical Software* **4**(3) (1978), 237–249.

W. S. Brown and J. F. Traub, On Euclid's Algorithm and the Theory of Subresultants. *Journal of the ACM* **18**(4) (1971), 505–514.

G. E. Collins, Polynomial remainder sequences and determinants. *The American Mathematical Monthly* **73** (1966), 708–712.

George E. Collins, Subresultants and Reduced Polynomial Remainder Sequences. *Journal of the ACM* **14**(1) (1967), 128–142.

George E. Collins, The Calculation of Multivariate Polynomial Resultants. *Journal of the ACM* **18**(4) (1971), 515–532.

G. E. Collins, Computer algebra of polynomials and rational functions. *The American Mathematical Monthly* **80** (1973), 725–755.

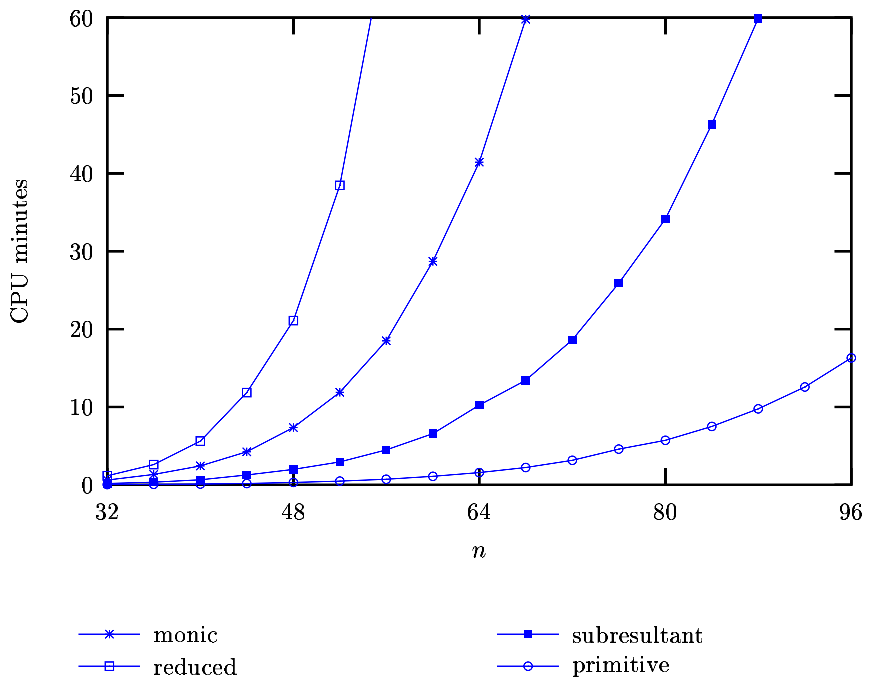

Fig. 4. Computation of non-normal polynomial remainder sequences for polynomials of degree $2n-1$ with coefficient length less than n and a degree jump of $\frac{n}{3}$ at degree $2n-\frac{n}{6}$, for $32 \le n \le 96$. Time is now measured in minutes.

Gene Cooperman, Sandra Feisel, Joachim von zur Gathen, and George Havas, Gcd of many integers. In *COCOON '99*, ed. T. Asano et al., Lecture Notes in Computer Science **1627**. Springer-Verlag, 1999, 310–317.

Leonhard Euler, Démonstration sur le nombre des points où deux lignes des ordres quelconques peuvent se couper. *Mémoires de l'Académie des Sciences de Berlin* **4** (1748), 1750, 234–248. Eneström 148. *Opera Omnia*, ser. 1, vol. 26, Orell Füssli, Zürich, 1953, 46–59.

Joachim von zur Gathen, Parallel algorithms for algebraic problems. *SIAM Journal on Computing* **13**(4) (1984), 802–824.

Joachim von zur Gathen and Jürgen Gerhard, *Modern Computer Algebra*. Cambridge University Press, 1999.

K. O. Geddes, S. R. Czapor, and G. Labahn, *Algorithms for Computer Algebra*. Kluwer Academic Publishers, 1992.

Paul Gordan, *Vorlesungen über Invariantentheorie. Erster Band: Determinanten.* B. G. Teubner, Leipzig, 1885. Herausgegeben von Georg Kerschensteiner.

Walter Habicht, Eine Verallgemeinerung des Sturmschen Wurzelzählverfahrens. *Commentarii Mathematici Helvetici* **21** (1948), 99–116.

M. W. Haskell, Note on resultants. *Bulletin of the New York Mathematical Society* **1** (1892), 223–224.

THOMAS W. HUNGERFORD, *Abstract Algebra: An Introduction.* Saunders College Publishing, Philadelphia PA, 1990.

C. G. J. JACOBI, De eliminatione variabilis e duabus aequationibus algebraicis. *Journal für die Reine und Angewandte Mathematik* **15** (1836), 101–124.

DONALD E. KNUTH, *The Art of Computer Programming, vol.2, Seminumerical Algorithms.* Addison-Wesley, Reading MA, 2nd edition, 1981.

L. KRONECKER, Die verschiedenen *Sturm*schen Reihen und ihre gegenseitigen Beziehungen. *Monatsberichte der Königlich Preussischen Akademie der Wissenschaften, Berlin* (1873), 117–154.

L. KRONECKER, Zur Theorie der Elimination einer Variabeln aus zwei algebraischen Gleichungen. *Monatsberichte der Königlich Preussischen Akademie der Wissenschaften, Berlin* (1881), 535–600. *Werke*, Zweiter Band, ed. K. HENSEL, Leipzig, 1897, 113–192. Reprint by Chelsea Publishing Co., New York, 1968.

THOMAS LICKTEIG AND MARIE-FRANÇOISE ROY, Cauchy Index Computation. *Calcolo* **33** (1997), 331–357.

R. LOOS, Generalized Polynomial Remainder Sequences. *Computing* **4** (1982), 115–137.

THOM MULDERS, A note on subresultants and the Lazard/Rioboo/Trager formula in rational function integration. *Journal of Symbolic Computation* **24**(1) (1997), 45–50.

ISAAC NEWTON, *Arithmetica Universalis, sive de compositione et resolutione arithmetica liber.* J. Senex, London, 1707. English translation as *Universal Arithmetick: or, A Treatise on Arithmetical composition and Resolution*, translated by the late Mr. Raphson and revised and corrected by Mr. Cunn, London, 1728. Reprinted in: DEREK T. WHITESIDE, *The mathematical works of Isaac Newton*, Johnson Reprint Co, New York, 1967, p. 4 ff.

DANIEL REISCHERT, Asymptotically Fast Computation of Subresultants. In *Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation ISSAC '97*, Maui HI, ed. WOLFGANG W. KÜCHLIN. ACM Press, 1997, 233–240.

V. STRASSEN, The computational complexity of continued fractions. *SIAM Journal on Computing* **12**(1) (1983), 1–27.

C. STURM, Mémoire sur la résolution des équations numériques. *Mémoires présentés par divers savants à l'Acadèmie des Sciences de l'Institut de France* **6** (1835), 273–318.

J. J. SYLVESTER, A method of determining by mere inspection the derivatives from two equations of any degree. *Philosophical Magazine* **16** (1840), 132–135. *Mathematical Papers* **1**, Chelsea Publishing Co., New York, 1973, 54–57.

RICHARD ZIPPEL, *Effective polynomial computation.* Kluwer Academic Publishers, 1993.