# On Computing Polynomial GCDs in Alternate Bases

Howard Cheng
Dept. of Mathematics and Computer Science
University of Lethbridge, Lethbridge, Canada
cheng@cs.uleth.ca

George Labahn
Symbolic Computation Group
School of Computer Science
University of Waterloo, Waterloo, Canada
glabahn@uwaterloo.ca

## ABSTRACT

In this paper, we examine the problem of computing the greatest common divisor (GCD) of univariate polynomials represented in different bases. When the polynomials are represented in Newton basis or a basis of orthogonal polynomials, we show that the well-known Sylvester matrix can be generalized. We give fraction-free and modular algorithms to directly compute the GCD in the alternate basis. These algorithms are suitable for computation in domains where growth of coefficients in intermediate computations are a central concern. In the cases of Newton basis and bases using certain orthogonal polynomials, we also show that the standard subresultant algorithm can be applied easily. If the degrees of the input polynomials is at most $n$ and the degree of the GCD is at least $n/2$, our algorithms outperform the corresponding algorithms using the standard power basis.

## Categories and Subject Descriptors

I.1.2 [**Symbolic and Algebraic Manipulation**]: Algorithms—*Algebraic algorithms*

## General Terms

Algorithms

## Keywords

Krylov Matrices, Euclidean Algorithm, Orthogonal Polynomials, Fraction-free arithmetic, Subresultants.

## 1. INTRODUCTION

In this paper we consider the problem of computing the greatest common divisor (GCD) of two polynomials

$$a(x) = a_0\omega_0(x) + \cdots + a_m\omega_m(x) \quad \text{and}$$
$$b(x) = b_0\omega_0(x) + \cdots + b_n\omega_n(x)$$

represented in non-standard bases $\{\omega_i(x)\}_{i=0,1,\ldots}$. Examples include polynomials given in terms of Newton bases or in terms of a basis of orthogonal polynomials. Such polynomials arise naturally in linear control theory [4, Sec 5.3-5.4], interpolation problems and rational interpolation problems.

Of course one can always convert polynomials in alternate bases into polynomials in the standard power basis, do computation of the GCD and then convert back to our alternate bases. We wish to avoid such conversions to both reduce computational cost and also to describe the intermediate computations in terms of the original bases.

There has been considerable work on manipulation of polynomials represented in alternate bases. Barnett made use of companion matrices and their so-called congenial matrix generalizations (colleague, comrade and confederate matrices) in order to give procedures for computing GCDs [3, 4] and later division [5] and Euclidean remainders [6]. Gemignani [19] used structured matrix techniques to improve on complexity costs associated to Barnett's algorithms. Later Diaz-Toca and Gonzalez-Vega [16, 17] made use of Bezoutian matrices in order to compute GCDs in alternate bases.

Our goal is to compute GCDs in exact arithmetic environments, more specifically, in exact arithmetic environments where coefficient growth is a concern. In the case of polynomials represented in standard power basis effective GCD algorithms include the well-known subresultant GCD [9, 14, 18, 22] and modular GCD algorithms [10]. It is these algorithms which we generalize in this paper.

Computation in exact environments gives additional reasons for avoiding conversion to standard power basis. For example, conversion can convert a polynomial coming from a simple computational domain (say with coefficients from an integral domain such as the integers) into one having coefficients from an algebraically more involved domain (for example the quotient field of the domain). This has a negative effect on fraction-free GCD methods such as the subresultant algorithm unless coefficient GCD operations are used to first remove the contents. In addition, conversion may introduce unnecessary coefficient growth. For example, in a Newtonian basis converting the input polynomials will require multiplying most of the interpolation points together. In the case of modular computation homomorphisms which are lucky (i.e. reduce to correct degrees of GCDs) in alternate bases may be unlucky homomorphisms in the standard power basis.

Our approach is to make use of structured linear systems, replacing Sylvester matrices by striped Krylov matrices. In the case of fraction-free arithmetic we do structured fraction-free elimination as done in the FFFG algorithm of [7]. Our modular procedures make use of the generalized subresultant procedure to determine unlucky homomorphisms and to give proper stopping criterion. Although linear algebra and subresultant algorithms are often less efficient than the modular counterparts, they often form a basis on which more efficient algorithms are developed.

The remainder of the paper is organized as follows. In the next section we introduce our basis along with the diophantine equations that are solved in GCD problems. Section 3 gives a linear algebraic formulation to the GCD problems with striped Krylov matrices as a matrix of coefficients of our linear system. Section 4 gives an algorithm for fraction-free computation which is based on the FFFG algorithm of [7] while section 5 gives a generalization of subresultants and the subresultant algorithm. Section 6 looks at modular computation including unlucky homomorphisms, normalization, and termination. The paper ends with a conclusion and topics for future research.

## 2. DIOPHANTINE EQUATIONS

Our approach can be informally described as follows. Let $f_1(x)$ and $f_2(x)$ be two polynomials in a specific alternate basis. We look to compute a sequence of polynomial pairs $u_i(z)$ and $v_i(z)$ such that

$$u_i(z) \cdot f_1(x) + v_i(z) \cdot f_2(x) = r_i(x) \tag{1}$$

with $u_i(z)$ and $v_i(z)$ polynomials in the standard power basis in a *special* variable $z$. Here $r_i(x)$ is a polynomial given in an alternate basis form and has decreasing degree as $i$ increases. The notion of using polynomials in special elements is taken from the approach used in [8].

More formally, let $\mathbb{D}$ be an integral domain with $\mathbb{F}$ its quotient field. Let $\mathcal{V}$ be an infinite dimensional vector space over $\mathbb{F}$ having a basis $(\omega_i)_{i=0,1,\ldots}$ with $(c_i)_{i=0,1,\ldots}$ its dual basis (i.e. a set of linear functionals on $\mathcal{V}$ satisfying $c_i(\omega_j) = \delta_{i,j}$). Thus every element $f$ of $\mathcal{V}$ can be written as

$$f = f_0 \cdot \omega_0 + f_1 \cdot \omega_1 + f_2 \cdot \omega_2 + \cdots \tag{2}$$

with $c_i(f) = f_i$, the $i$-th coefficient of $f$ in the chosen basis. We define the $\mathcal{V}$-*degree* of a nontrivial element $f$ of $\mathcal{V}$ in the obvious way by

$$\deg_{\mathcal{V}}(f) = n \quad \text{iff} \quad c_n(f) \neq 0 \text{ and } c_i(f) = 0 \text{ for all } i > n.$$

We assume that we have a *special element* $z$ that acts on $\mathcal{V}$ via a special multiplication rule

$$c_i(z \cdot f) = c_{i,i-1} \cdot c_{i-1}(f) + c_{i,i} \cdot c_i(f) + c_{i,i+1} \cdot c_{i+1}(f), \tag{3}$$

with $c_{i,j} \in \mathbb{D}$ and $c_{i,i+1} \neq 0$. We also define $c_{i,j} = 0$ if $j \notin \{i-1, i, i+1\}$. This special rule can be viewed as a type of Leibniz chain rule. The special rule allows us to define a multiplication $p(z) \cdot f$ for any polynomial $p \in \mathbb{F}[z]$ and $f \in \mathcal{V}$, making $\mathcal{V}$ an infinite dimensional module over $\mathbb{F}[z]$. Note that this special rule is an extension of that used in [8] (where $c_{i,i+1} = 0$). This extension is important since it allows us to include polynomials represented in terms of orthogonal polynomial bases.

Example 2.1. *Let $\mathcal{V}$ be the polynomial ring $\mathbb{F}[x]$ with basis $(x^i)_{i=0,1,\ldots}$ and let $c_{i,j} = \delta_{i-1,j}$. Then the special multiplication rule is simply the standard multiplication by $x$.*

Example 2.2. *Let $\mathcal{V}$ be the space of all formal Newton series in $x$ with respect to the given knots $x_0, x_1, \ldots$ with basis elements $\omega_i = \prod_{j=0}^{i-1}(x - x_j)$. Then $c_i$ is the $i$-th divided difference $[x_0, \ldots, x_i]$. It is easy to verify that for these linear functionals the special multiplication rule (3) holds, with $c_{i,j} = \delta_{i,j} \cdot x_i + \delta_{i-1,j}$, $i > 0$, and $c_{0,0} = x_0$.*

In all the previous examples we have $c_{i,i+1} = 0$ for all $i$. In the following we give an important example where this is not the case.

Example 2.3 (Orthogonal Polynomials). *Suppose we choose the set of Chebyshev polynomials $T_i(x)$ for $i = 0, 1, \ldots$ as a basis for $\mathcal{V} = \mathbb{F}[x]$. Then $z = 2x$ is a special element with a rule given by [1]*

$$z \cdot T_i(x) = T_{i-1}(x) + T_{i+1}(x). \tag{4}$$

*Similarly, if the basis consists of generalized Laguerre polynomials $L_i^{(\alpha)}(x)$ then $z = x$ is a special element with the rule*

$$z \cdot L_i^{(\alpha)}(x) = -(i+\alpha)L_{i-1}^{(\alpha)}(x) + (2i+\alpha+1)L_i^{(\alpha)}(x) - (i+1)L_{i+1}^{(\alpha)}(x). \tag{5}$$

*In general all the well known orthogonal polynomials $\{P_i(x)\}$ have special rules with a special element $z = ax + b$ for some constants $a, b$ given by*

$$z \cdot P_i(x) = c_{i+1,i} \cdot P_{i+1}(x) + c_{i,i} \cdot P_i(x) + c_{i-1,i} \cdot P_{i-1}(x) \tag{6}$$

*with $c_{i,j} \in \mathbb{D}$ and with $c_{i+1,i}, c_{i-1,i} \neq 0$.*

We make the following assumptions on $\mathcal{V}$ and the basis $(\omega_u)$:

1. $\mathcal{V} = \mathbb{D}[x]$;
2. $\deg_x \omega_i = i$, so that the $\mathcal{V}$-degree is simply the standard degree;
3. $z = ax + b$ for some $a, b \in \mathbb{F}$.

These assumptions are satisfied by the previous examples. To simplify notations, we will use deg to denote either $\mathcal{V}$-degree or the standard degree. With these assumptions, one can see that the property $\deg(z \cdot f) = \deg(f) + 1$ holds.

Remark 2.4. *We summarize a number of common choices of basis in Table 1, including well known classical orthogonal polynomials as well as some discrete orthogonal polynomials. We include the corresponding coefficients $c_{i,j}$ for the three term recurrences [1]. Other bases include the Moak polynomials and the Kravchouk polynomials. Note that $c_{i,i-1} = 1$ in many cases. Although the coefficients for Legendre and ultraspherical polynomials are fractions if $\mathbb{D} = \mathbb{Z}$, we can still consider them in our framework if $\mathbb{D} = \mathbb{F}[t]$.*

In this paper we will study the following linear diophantine problem.

Definition 2.5 (Linear Diophantine Problem). *Let $f_1(x)$, $f_2(x) \in \mathcal{V}$, $\tau$ be a positive integer and $(n_u, n_v)$ be degree bounds. Determine polynomials $u(z)$ and $v(z)$ in $z$, with $\deg_z u(z) \leq n_u$ and $\deg_z v(z) \leq n_v$, such that*

$$\deg(u(z) \cdot f_1(x) + v(z) \cdot f_2(x)) < \tau. \tag{7}$$

*In this case, $[u(z), v(z)]$ will be referred to as a diophantine solution of type $(\tau, n_u, n_v)$.*

**Table 1: Common polynomial bases.**

| Basis | $\omega_i(x)$ | $z$ | $c_{i,i+1}$ | $c_{i,i}$ | $c_{i,i-1}$ |
|---|---|---|---|---|---|
| Standard | $x^i$ | $x$ | 0 | 0 | 1 |
| Newton | $\prod_{j=0}^{i-1}(x-x_j)$ | $x$ | 0 | $x_i$ | 1 |
| Chebyshev | $T_i(x)$ | $2x$ | 1 | 0 | 1 |
| Chebyshev | $U_i(x)$ | $2x$ | 1 | 0 | 1 |
| Shifted Chebyshev | $T_i^*(x)$ | $4x-2$ | 1 | 0 | 1 |
| Shifted Chebyshev | $U_i^*(x)$ | $4x-2$ | 1 | 0 | 1 |
| Hermite | $H_i(x)$ | $2x$ | $2i+2$ | 0 | 1 |
| Generalized Laguerre | $L_i^{(\alpha)}(x)$ | $x$ | $-i-\alpha-1$ | $2i+\alpha+1$ | $-i$ |
| Legendre | $P_i(x)$ | $x$ | $\frac{i+1}{2i+3}$ | 0 | $\frac{i}{2i-1}$ |
| Ultraspherical | $C_i^{(\alpha)}(x)$ | $2x$ | $\frac{i+2\alpha}{i+\alpha+1}$ | 0 | $\frac{i}{i+\alpha-1}$ |
| Meixner | $m_i(x;a,b)$ | $(b-1)x$ | $(i+1)(i+a)$ | $-i-bi-ab$ | $b$ |
| Charlier | $c_i(x;a)$ | $x$ | $-i-1$ | $i+a$ | $-a$ |

## 3. ASSOCIATED LINEAR SYSTEMS

Let us have a closer look at the underlying system of linear equations that results from a diophantine equation of the form (7). Notice first that we may rewrite the special multiplication rule (3) in terms of linear algebra. Let $\mathbf{C}_\sigma = (c_{i,j})_{i,j=\sigma-1,\ldots,0}$ $(\sigma \geq 0)$. Furthermore, for each $f \in \mathcal{V}$ and nonnegative integer $\sigma$ we associate a vector of coefficients

$$\mathbf{F}_\sigma = [c_{\sigma-1}(f), \ldots, c_0(f)]^T. \qquad (8)$$

Note that we begin our row enumeration with index $\sigma-1$ toward 0. Provided that $\deg f < \sigma$, the special multiplication rule can be interpreted in matrix form as

$$\mathbf{C}_\sigma \cdot \mathbf{F}_\sigma = [c_{\sigma-1}(z\cdot f), \ldots, c_0(z\cdot f)]^T \qquad (9)$$

and more generally

$$p(\mathbf{C}_\sigma)\cdot \mathbf{F}_\sigma = [c_{\sigma-1}(p(z)\cdot f), \ldots, c_0(p(z)\cdot f)]^T \qquad (10)$$

for any polynomial $p(z) \in \mathbb{F}[z]$ and for any nonnegative integer $\sigma \geq \deg f + \deg_z p(z)$. We now give a generalization of the Sylvester matrix which allows us to formulate (7) as a linear system of equations.

Definition 3.1. *Let $n_i = \deg f_i(x)$, $i = 1, 2$. The* striped Krylov matrix *of $f_1(x)$ and $f_2(x)$ is the matrix*

$$\mathbf{K}(f_1(x), f_2(x)) = [\mathbf{C}_\sigma^{n_2-1}\cdot\mathbf{F}_{1,\sigma}, \cdots, \mathbf{F}_{1,\sigma}, \mathbf{C}_\sigma^{n_1-1}\cdot\mathbf{F}_{2,\sigma}, \cdots, \mathbf{F}_{2,\sigma}], \qquad (11)$$

*where $\sigma = n_1 + n_2$. When the context is clear, we write $\mathbf{K} = \mathbf{K}(f_1(x), f_2(x))$. We also define the matrix $\mathbf{K}_{m,n}$ to be the matrix*

$$\mathbf{K}_{m,n} = [\mathbf{C}_\sigma^m\cdot\mathbf{F}_{1,\sigma}, \cdots, \mathbf{F}_{1,\sigma}, \mathbf{C}_\sigma^n\cdot\mathbf{F}_{2,\sigma}, \cdots, \mathbf{F}_{2,\sigma}], \qquad (12)$$

*$\mathbf{K}_{m,n}^\tau$ to be the matrix consisting of the rows indexed by $\sigma-1, \ldots, \tau$ in $\mathbf{K}_{m,n}$, and $\mathbf{K}_{m,n,\tau}$ to be the matrix consisting of rows $\tau-1, \ldots, 0$.*

When the matrix $C_\sigma$ is defined as in Example 2.1, the striped Krylov matrix is simply the well-known Sylvester matrix.

The linear diophantine problem (7) can now be stated as a linear algebra problem:

$$\mathbf{K}_{n_u,n_v}^\tau \cdot [u_{n_u}, \cdots, u_0, v_{n_v}, \cdots v_0]^T = \mathbf{0}. \qquad (13)$$

Finally, we extend the definition of the determinant polynomial of a matrix to our framework.

Definition 3.2. *Let $\mathbf{M}$ be an $\ell \times k$ matrix with $k \leq \ell$. The* determinant polynomial *of $\mathbf{M}$ is*

$$\mathrm{detpol}(\mathbf{M}) = \det(\mathbf{M}^{(\ell-k)})\cdot\omega_{\ell-k} + \cdots + \det(\mathbf{M}^{(0)})\cdot\omega_0,$$

*where $\mathbf{M}^{(j)}$ is the submatrix of $\mathbf{M}$ consisting of the first $k-1$ rows and row $j$ (indexed from $\ell-1$ to $0$).*

*Let $f_i(x) \in \mathcal{V}$ $(1 \leq i \leq m)$. We also define the determinant polynomial*

$$\mathrm{detpol}(f_1(x), \ldots, f_m(x)) = \mathrm{detpol}([\mathbf{F}_{1,\sigma}, \cdots, \mathbf{F}_{m,\sigma}])$$

*where $\sigma = 1 + \max_{1\leq i\leq m} \deg f_i(x)$.*

When $\omega_i = x^i$, this definition is identical to the usual definition of determinant polynomial [22]. The determinant polynomial represents the coefficients in the last nonzero column if one performs fraction-free Gaussian elimination on the matrix. The determinant polynomial of $\mathbf{K}_{m,n}$ gives the polynomial $u(z)\cdot f_1(x) + v(z)\cdot f_2(z)$ associated with a diophantine solution $[u(z), v(z)]$ of type $(\tau, m, n)$ for some bound $\tau$.

## 4. FRACTION-FREE COMPUTATION

From the assumptions made in the previous section, it is possible to treat the polynomials in the diophantine solution as polynomials in the indeterminate $x$ (since the set $\{z^i\}_{i=0,1,\ldots}$ is a polynomial basis of $\mathbb{F}[x]$). This in turn allows a number of well-known facts on the Sylvester matrix (and its submatrices) to be easily extended to striped Krylov matrices. In particular, the following fact is used to develop a fraction-free algorithm in this section (see, for example, [3, 17, 18]).

Theorem 4.1. *Let $n_1 = \deg f_1(x)$, $n_2 = \deg f_2(x)$ with $n_1 \geq n_2$. If $d = \deg\gcd(f_1(x), f_2(x))$, then* rank $\mathbf{K} = n_1 + n_2 - d$. *The coefficients of a GCD of $f_1(x)$ and $f_2(x)$ can be obtained by performing Gaussian elimination on $\mathbf{K}$ and selecting the coefficients of the last nonzero column.*

Our goal is to perform fraction-free Gaussian elimination [2] on $\mathbf{K}$ without explicitly constructing the matrix. A similar technique was used in [8] to perform fraction-free computation of the so-called *order basis*, which represents elimination on low-order coefficients. By taking the reciprocal of polynomials in the standard power basis, the coefficients are reversed and a fraction-free algorithm for the polynomial GCD is obtained. However, the same technique cannot be used here because one cannot easily reverse the coefficients in alternate bases.

In our case, we perform operations to eliminate the leading coefficients of the polynomials $f_i(x)$, so that the operations correspond to those performed by fraction-free Gaussian elimination on $\mathbf{K}_{m,n,\tau}$ for increasing values of $m$ and $n$. Starting with $(m, n, \tau) = (0, n_1 - n_2, n_1 + 1)$, we perform fraction-free Gaussian elimination on the columns of $\mathbf{K}_{m,n,\tau}$ by keeping only the last column of each stripe. We denote these two intermediate results as $r_1(x)$ and $r_2(x)$.

We start by setting $r_1(x) = f_1(x)$ and $r_2(x) = z^{n_1-n_2} \cdot f_2(x)$. We also keep track of $p_i(z)$, $q_i(z)$ such that

$$p_i(z) \cdot f_1(x) + q_i(z) \cdot f_2(x) = r_i(x). \tag{14}$$

Thus, $p_1(z) = 1$, $q_2(z) = z^{n_1-n_2}$, and $p_2(z) = q_1(z) = 0$. At each step, we choose one of the polynomials, say $r_\pi(x)$, as a pivot to eliminate the leading coefficient of the other. If the non-pivot polynomial becomes identically zero after this step, then $r_\pi(x)$ is a GCD of $f_1(x)$ and $f_2(x)$. Otherwise, we must consider the next column in stripe $\pi$ of the underlying striped Krylov matrix. Assuming that $z$ divides $p_\pi(z)$ and $q_\pi(z)$, this column can be obtained from $z^{-1} \cdot r_\pi(x)$ followed by appropriate adjustment to obtain the appropriate scalar multiple. On the other hand, if either $p_\pi(z)$ or $q_\pi(z)$ has a nonzero constant coefficient, the underlying striped Krylov matrix must be expanded implicitly by multiplying all intermediate results by $z$ and adjusting both $r_1(x)$ and $r_2(x)$ appropriately to ensure that their coefficients are the same as those obtained by fraction-free Gaussian elimination on the expanded striped Krylov matrix. The problem is then reduced to the previous case.

Let us first state and prove some key results required to give our fraction-free algorithm. A key step in the new fraction-free algorithm described above is the expansion of the striped Krylov matrix. Since the intermediate results $r_1(x)$ and $r_2(x)$ are obtained from fraction-free Gaussian elimination, it follows that they can be represented as determinant polynomials of submatrices of $\mathbf{K}_{m,n}$, at least up to sign. For convenience, we define

$$C_i^j = \prod_{t=i}^{j} c_{t,t-1} \tag{15}$$

which are quantities occurring frequently in our expressions. The following theorem describes the relationship needed to adjust the coefficients after multiplying the results by $z$.

Theorem 4.2. *Let $n_i = \deg f_i(x)$, $i = 1, \ldots, m$, and $n = 1 + \max_{1 \le i \le m} n_i$. If $m < n$, then*

$$\begin{aligned}&\operatorname{detpol}(z \cdot f_1(x), \ldots, z \cdot f_m(x))\\ &\quad = C_{n-m+2}^{n} \cdot z \cdot \operatorname{detpol}(f_1(x), \ldots, f_m(x)).\end{aligned} \tag{16}$$

Proof. For arbitrary $\sigma \in \mathbb{N}$, let $\mathbf{M}_\sigma = [F_{1,\sigma}, \ldots, F_{m,\sigma}]$. Then $\operatorname{detpol}(\mathbf{M}_n) = \operatorname{detpol}(f_1(x), \ldots, f_m(x))$, and

$$\operatorname{detpol}(z \cdot f_1(x), \ldots, z \cdot f_m(x)) = \operatorname{detpol}(\mathbf{C}_{n+1} \cdot \mathbf{M}_{n+1}).$$

Let $\mathbf{C}$ be $\mathbf{C}_{n+1}$ with the first column removed. Since the first row of $\mathbf{M}_{n+1}$ is zero, it follows that $\operatorname{detpol}(\mathbf{C}_{n+1} \cdot \mathbf{M}_{n+1}) = \operatorname{detpol}(\mathbf{C} \cdot \mathbf{M}_n)$. We have

$$\operatorname{detpol}(\mathbf{C} \cdot \mathbf{M}_n) = \sum_{i=0}^{n+1-m} \det((\mathbf{C} \cdot \mathbf{M}_n)^{(i)}) \cdot \omega_i, \tag{17}$$

and

$$\begin{aligned} z \cdot \operatorname{detpol}(\mathbf{M}_n) &= z \cdot \sum_{i=0}^{n-m} \det(\mathbf{M}_n^{(i)}) \cdot \omega_i \\ &= \sum_{i=0}^{n+1-m} \left( \sum_{t=i-1}^{i+1} c_{i,t} \det(\mathbf{M}_n^{(t)}) \right) \cdot \omega_i. \end{aligned} \tag{18}$$

Since determinants are multi-linear, $\sum_{t=i-1}^{i+1} c_{i,t} \det(\mathbf{M}_n^{(t)})$ can be written as the determinant of a single matrix $\mathbf{M}'$ whose last row coincides with the last row of $(\mathbf{C} \cdot \mathbf{M}_n)^{(i)}$ while the remaining rows are the same as those of $\mathbf{M}_n$. Let $\mathbf{C}'$ be the matrix obtained by taking rows $n, \ldots, n-m+2$ of $\mathbf{C}$ and adding the row $[0, \ldots, 1]$. Then $\mathbf{C}' \cdot \mathbf{M}' = (\mathbf{C} \cdot \mathbf{M}_n)^{(i)}$, and hence

$$\begin{aligned} \det((\mathbf{C} \cdot \mathbf{M}_n)^{(i)}) &= \det(\mathbf{C}') \cdot \det(\mathbf{M}') \\ &= C_{n-m+2}^{n} \cdot \sum_{t=i-1}^{i+1} c_{i,t} \det(\mathbf{M}_n^{(t)}). \end{aligned} \tag{19}$$

The last equality follows from the fact that $\mathbf{C}$ is lower triangular with diagonal entries $c_{t,t-1}$ $(t = n, \ldots, 0)$ as $c_{i,j} = 0$ if $j < i - 1$. The equality (16) now follows from (17), (18), and (19). □

Next, we give a result related to the correctness of our termination criteria.

Theorem 4.3. *Let $k$ be the largest value such that*

$$\operatorname{rank} \mathbf{K}_{n_2-1-k, n_1-1-k} < n_1 + n_2 - 2k \quad (0 \le k \le n_2 - 1).$$

*Then the last nonzero column resulting from applying Gaussian elimination on $\mathbf{K}_{n_2-1-k,n_1-1-k}$ gives a GCD of $f_1(x)$ and $f_2(x)$. If no such $k$ exists, then* $\gcd(f_1(x), f_2(x)) = 1$.

Proof. If $k = 0$, the result is immediate from Theorem 4.1. On the other hand, if $k > 0$, consider the matrix $\mathbf{K}_{n_2-1-k,n_1-1-k}(z^k \cdot f_1(x), z^k \cdot f_2(x))$. Since the results of performing fraction-free Gaussian elimination on a matrix can be represented as determinant polynomials of its submatrices, Theorem 4.2 implies that the results obtained from $\mathbf{K}_{n_2-1-k,n_1-1-k}(z^k \cdot f_1(x), z^k \cdot f_2(x))$ are the same as those obtained from $\mathbf{K}_{n_2-1-k,n_1-1-k}(f_1(x), f_2(x))$ up to a nonzero multiplicative constant. In particular, a column is zero in the reduced $\mathbf{K}_{n_2-1-k,n_1-1-k}(z^k \cdot f_1(x), z^k \cdot f_2(x))$ if and only if the corresponding column is zero in the reduced $\mathbf{K}_{n_2-1-k,n_1-1-k}(f_1(x), f_2(x))$. Now, the striped Krylov matrix $\mathbf{K}(f_1(x), f_2(x))$ can be constructed by combining $\mathbf{K}_{n_2-1-k,n_1-1-k}(z^k \cdot f_1(x), z^k \cdot f_2(x))$ and $\mathbf{K}_{n_2-1-k,n_1-1-k}(f_1(x), f_2(x))$ (and removing duplicated columns). Therefore, the last nonzero column in the reduced $\mathbf{K}(f_1(x), f_2(x))$ is the same as the last nonzero column in the reduced $\mathbf{K}_{n_2-1-k,n_1-1-k}(f_1(x), f_2(x))$ up to a nonzero multiplicative constant. The result now follows from Theorem 4.1.

If no such $k$ exists, then $\mathbf{K}(f_1(x), f_2(x))$ must have full rank. It follows from Theorem 4.1 that $\deg \gcd(f_1(x), f_2(x)) = 0$. □

The last ingredient relates to the choice of pivot. We keep track of a vector $\mu = (\mu_1, \mu_2)$ such that $\mu_i$ is the number of times $r_i(x)$ has been chosen as a pivot. If $\deg r_1(x) \ne \deg r_2(x)$, there is only one choice of $\pi$ such that $\deg r_\pi(x) \ge$

$\deg r_i(x)$ for $i = 1, 2$. Otherwise, the choice of $\pi$ should satisfy

$$\pi = \begin{cases} 1 & -\mu_1 \geq n_1 - n_2 - \mu_2, \\ 2 & \text{otherwise.} \end{cases} \tag{20}$$

This choice of pivots ensures that the underlying striped Krylov matrix is of the form $\mathbf{K}_{n_2-1-k,n_1-1-k}$ if possible, so that Theorem 4.3 can be applied. This is in fact identical to the concept of following the "closest normal path" in the Padé table [7, 8]. The well-known structure of the Padé table also ensures that if $\deg \gcd(f_1(x), f_2(x)) > 0$, there exists a $k \geq 0$ satisfying the hypothesis of Theorem 4.3.

**Algorithm 1** Fraction-free algorithm to compute GCD in alternate basis.

```
Input: $f_1(x), f_2(x) \in \mathbb{D}[x]$.
Output: $g(x) = \gcd(f_1(x), f_2(x)) \in \mathbb{D}[x]$.
  $[r_1, p_1, q_1] \leftarrow [f_1, 1, 0]$
  $[r_2, p_2, q_2] \leftarrow [z^{n_1-n_2} \cdot f_2, 0, z^{n_1-n_2}]$
  $[\mu, s, d] \leftarrow [(0, 0), 0, 1]$
  while $r_1 \neq 0$ and $r_2 \neq 0$ do
    choose $\pi$ according to (20)
    if $p_\pi(z)$ or $q_\pi(z)$ has a nonzero constant coefficient then
      $[r_i, p_i, q_i] \leftarrow C^{n_1+s+1}_{s+n_1-\mu_1-\mu_2+2} \cdot z \cdot [r_i, p_i, q_i], \quad i = 1, 2$
      $[d, s] \leftarrow [C^{n_1+s+1}_{s+n_1-\mu_1-\mu_2+2} \cdot d, s + 1]$
    end if
    $\lambda_i \leftarrow \operatorname{lcoeff}(r_i(x))$
    $\rho \leftarrow 3 - \pi$
    $\gamma = \begin{cases} \operatorname{coeff}(q_1(z), z^{n_1-n_2+s+1-\mu_2}) & \pi = 1, \\ \operatorname{coeff}(p_2(z), z^{s+1-\mu_1}) & \pi = 2. \end{cases}$
    $[r_\rho, p_\rho, q_\rho] \leftarrow (\lambda_\pi \cdot [r_\rho, p_\rho, q_\rho] - \lambda_\rho \cdot [r_\pi, p_\pi, q_\pi]) / d$
    $[r_\pi, p_\pi, q_\pi] \leftarrow (\lambda_\pi z^{-1} \cdot [r_\pi, p_\pi, q_\pi] - \gamma \cdot [r_\rho, p_\rho, q_\rho]) / d$
    $[d, \mu_\pi] \leftarrow [\lambda_\pi, \mu_\pi + 1]$
  end while
  $g(x) = r_1$ if $r_1 \neq 0$, or $r_2$ otherwise.
```

The complete algorithm is given in Algorithm 1. We note that the division of $z$ cannot introduce fractions (as the cofactors $p_i(z), q_i(z)$ have coefficients in $\mathbb{D}$), and that it can be performed in $O(\deg r_\pi(x))$ operations because of (3). Also, this algorithm can be easily modified to return the cofactors $u(z)$ and $v(z)$ with

$$g(x) = u(z) \cdot f_1(x) + v(z) \cdot f_2(x).$$

As in Bareiss's fraction-free Gaussian elimination algorithm, the known divisor is simply the leading coefficient of the pivot polynomial in the last step, except that adjustments have to be made according to Theorem 4.2 when the underlying matrix is expanded. The fact that no fractions are introduced during the algorithm can be proved in the same way as in the FFFG elimination algorithm. We refer the reader to [7] for more details.

## 5. SUBRESULTANT ALGORITHM FOR SPECIAL CASES

In this section, we derive an analogue of subresultants, polynomial remainder sequences (PRS), and the fundamental theorem of PRS. Our development closely follows the previous works [9, 14, 18, 22, 23, 24].

Let $R_1(x), \ldots, R_k(x)$ be a *pseudo-remainder sequence* defined by

$$\alpha_i R_{i-1}(x) = Q_i(z) \cdot R_i(x) + \beta_i R_{i+1}(x), \quad 1 < i \leq k,$$
$$\alpha_k R_{k-1}(x) = Q_k(z) \cdot R_k(x).$$

Let $n_i = \deg R_i(x)$, $\delta_i = n_{i-1} - n_i$, $\gamma_i = \delta_i + \delta_{i+1}$, and $r_i = \operatorname{lcoeff}(R_i(x))$. In order to perform pseudo-division so that each $Q_i(z)$ and $R_{i+1}(x)$ have coefficients in $\mathbb{D}$, we need to set

$$\alpha_i = \left( \prod_{m=1}^{\delta_i} C^{n_i+m}_{n_i+1} \right) r_i^{\delta_i+1} = \left( \prod_{t=n_i+1}^{n_{i-1}} c^{n_{i-1}+1-t}_{t,t-1} \right) r_i^{\delta_i+1}. \tag{21}$$

As in classical subresultant theory, the *pseudo-remainder* can be represented as a determinant polynomial:

$$\operatorname{prem}(R_{i-1}(x), R_i(x)) = (-1)^{\delta_i+1} \operatorname{detpol}(R_{i-1}(x), z^{\delta_i} \cdot R_i(x), \cdots, R_i(x)). \tag{22}$$

Given two polynomials $A(x)$ and $B(x)$ of degrees $n_a$ and $n_b$, we also define the *j-th subresultant* in an analogous manner:

$$S(j, A(x), B(x)) = \operatorname{detpol}(z^{n_b-j-1} A(x), \ldots, A(x), z^{n_a-j-1} B(x), \ldots, B(x)). \tag{23}$$

In the generalization of the subresultant theory to Ore polynomials, Li [21] noted that a key fact needed in the subresultant theory is that

$$\operatorname{prem}(x^k \cdot A(x), x^k \cdot B(x)) = x^k \cdot \operatorname{prem}(A(x), B(x)).$$

While this is not true in the Ore polynomial case, Li showed that the difference between the two quantities can be expressed as a linear combination of other columns used in defining the subresultant. In our case, the difference between the quantities is simply a multiplicative constant.

Corollary 5.1. *Suppose that $A(x)$ and $B(x)$ are polynomials of degrees $n_a$ and $n_b$, respectively, with $n_a \geq n_b$. Then*

$$\operatorname{prem}(z^k \cdot A(x), z^k \cdot B(x)) = \left( \prod_{m=1}^{k} C^{n_a+m}_{n_b+m} \right) z^k \cdot \operatorname{prem}(A(x), B(x)). \tag{24}$$

Proof. The result follows immediately from the determinant polynomial representation of the pseudo-remainders and Theorem 4.2 by induction on $k$. □

With this result, we are able to express the relationship between subresultants and the elements of a PRS. We now state a number of results that are analogous to those for the classical subresultant theory. The proofs are omitted as they are similar but tedious. See, for example, [18, Section 7.3].

Lemma 5.2. *Suppose that $A(x) = Q(z) \cdot B(x) + R(x)$ where $\deg R(x) = k$ ($n_a \geq n_b > k$). Let $b = \operatorname{lcoeff}(B(x))$*

*and* $r = \operatorname{lcoeff}(R(x))$. *Then*

$$S(j, A(x), B(x)) = (-1)^{(n_a-j)(n_b-j)} \left( \prod_{\ell=1}^{n_b-j-1} \prod_{m=1}^{\ell} C_{n_b+m}^{n_a+m} \right) \cdot$$

$$\begin{cases} C_{k-j,n_a-j-1,n_b} \cdot b^{n_a-k} S(j, B(x), R(x)) & 0 \le j < k \\ C_{1,n_a-k-1,n_b} \cdot C_{1,n_b-k-1,k} \cdot b^{n_a-k} r^{n_b-k-1} R(x) & j = k \\ C_{1,n_a-n_b,n_b} \cdot b^{n_a-n_b+1} R(x) & j = n_b - 1 \\ 0 & \textit{otherwise.} \end{cases} \tag{25}$$

*with* $C_{i,j,k} = \prod_{m=i}^{j} C_{k+1}^{k+m}$.

Lemma 5.3.

$$S(j, R_{i-1}(x), R_i(x)) \cdot \alpha_i^{n_i-j} =$$

$$(-1)^{(n_{i-1}-j)(n_i-j)} \cdot \left( \prod_{\ell=1}^{n_i-j-1} \prod_{m=1}^{\ell} C_{n_i+m}^{n_{i-1}+m} \right) \cdot$$

$$\begin{cases} C_{n_{i+1}-j,n_{i-1}-j-1,n_i} \cdot r_i^{\gamma_i} \beta_i^{n_i-j} S(j, R_i(x), R_{i+1}(x)) & 0 \le j \le n_{i+1} \\ C_{1,\gamma_i-1,n_i} \cdot C_{1,\delta_{i+1}-1,n_{i+1}} \cdot r_i^{\gamma_i} r_{i+1}^{\delta_{i+1}-1} \beta_i^{\delta_{i+1}} R_{i+1}(x) & j = n_{i+1} \\ C_{1,\delta_i,n_i} \cdot r_i^{\delta_i+1} \beta_i R_{i+1}(x) & j = n_i - 1 \\ 0 & \textit{otherwise.} \end{cases} \tag{26}$$

Theorem 5.4 (Fundamental Theorem of PRS).

$$S(j, R_1(x), R_2(x)) = \begin{cases} \eta_i R_i(x) & j = n_{i-1} - 1 \\ \tau_i R_i(x) & j = n_i \\ 0 & \textit{otherwise} \end{cases}, \tag{27}$$

*where*

$$\eta_i = (-1)^{\phi_i} r_{i-1}^{1-\delta_i} \prod_{p=1}^{i-1} \left[ \left( \prod_{\ell=1}^{n_p-n_{i-1}} \prod_{m=1}^{\ell} C_{n_p+m}^{n_{p-1}+m} \right) \cdot \right.$$
$$\left. \left( \prod_{m=n_{p+1}-n_{i-1}+1}^{n_{p-1}-n_{i-1}} C_{n_p+1}^{n_p+m} \right) r_p^{\gamma_p} \left( \frac{\beta_p}{\alpha_p} \right)^{n_p-n_{i-1}+1} \right]$$

$$\tau_i = (-1)^{\sigma_i} r_i^{\delta_i-1} \left( \prod_{p=1}^{i-2} \prod_{\ell=1}^{n_p-n_i-1} \prod_{m=1}^{\ell} C_{n_p+m+1}^{n_{p-1}+m} \right) \left( \prod_{m=1}^{\delta_i-1} C_{n_i+1}^{n_i+m} \right) \cdot$$
$$\prod_{p=1}^{i-1} \left[ \left( \prod_{m=n_{p+1}-n_i}^{n_{p-1}-n_i-1} C_{n_p+1}^{n_p+m} \right) \left( \frac{\beta_p}{\alpha_p} \right)^{n_p-n_i} r_p^{\gamma_p} \right]$$

$$\phi_i = \sum_{p=1}^{i-1} (n_p - n_{i-1} + 1)(n_{p-1} - n_{i-1} + 1)$$

$$\sigma_i = \sum_{p=1}^{i-1} (n_p - n_i)(n_{p-1} - n_i).$$

The Fundamental Theorem of PRS shows how the elements in a PRS are related to the subresultants, and is usually used to show that a certain choice of $\alpha_i$ and $\beta_i$ lead to polynomial remainder sequences whose elements have coefficients in $\mathbb{D}$. However, the introduction of the factors of $c_{t,t-1}$ makes it difficult to choose $\alpha_i$ and $\beta_i$ in such a way to make $\eta_i = 1$, even when $c_{t,t-1} = \kappa$ for some constant $\kappa$.

In the special case where $c_{t,t-1} = 1$ for all $t > 0$, however, the products of $c_{t,t-1}$ can be ignored and we obtain a Fundamental Theorem of PRS that is identical to that in the classical subresultant theory. In this case, the choice of $\alpha_i$ and $\beta_i$ in the reduced and subresultant PRS [9, 14] can be used without modification, except that pseudo-division is performed in the alternate basis. This important special case is applicable to a number of choices of polynomial basis as shown in Remark 2.4.

Example 5.5. *Let*

$$f_1(x) = T_8(x) + T_7(x) + 2\,T_6(x) - 2\,T_4(x) - T_3(x) - 2\,T_2(x) + 2\,T_1(x) + T_0(x),$$
$$f_2(x) = 3\,T_6(x) + T_5(x) + 2\,T_4(x) - T_2(x) + 2\,T_1(x) + T_0(x),$$

*where $T_i(x)$ are the Chebyshev polynomials. Applying the subresultant algorithm directly in the given basis yields the following PRS (for compactness, we simply give a list of coefficients):*

$$R_1(x) = f_1(x), \quad R_2(x) = f_2(x)$$
$$R_3(x) = [40, 98, 60, 92, -10, -14]$$
$$R_4(x) = [596, -280, 584, -260, -28]$$
$$R_5(x) = [-7920, -3064, -1224, -1984]$$
$$R_6(x) = [16496, -125424, 1376]$$
$$R_7(x) = [-2052544, -246944]$$
$$R_8(x) = [6927616].$$

*If we first convert $f_1(x)$ and $f_2(x)$ into the standard power basis and apply the subresultant algorithm, we get the PRS:*

$$R_1(x) = [128, 64, -192, -112, 48, 52, 16, -2, 0]$$
$$R_2(x) = [96, 16, -128, -20, 36, 7, 1]$$
$$R_3(x) = [20971520, 25690112, -18350080, -19660800, 327680, -262144]$$
$$R_4(x) = [-39996882944, -9395240960, -30198988800, 4865392640, -134217728]$$
$$R_5(x) = [17008070492160, 3289944948736, -12098928872832, -579820584960]$$
$$R_6(x) = [283399122059264, -1077383956267008, -129879811031040]$$
$$R_7(x) = [70524874828808192, 8484931231547392]$$
$$R_8(x) = [238031073273970688].$$

*In this case, we see clearly that performing the subresultant directly in the alternate basis reduced unwanted coefficient growth.*

## 6. MODULAR ALGORITHMS

In this section, we describe a modular algorithm that is an analogue to the standard power basis case [10]. In order to obtain a modular algorithm, we must resolve three issues: the detection of unlucky primes, normalization, and termination.

We first examine the issue of the detection of unlucky primes. Let $\phi_p$ be the modular reduction that maps polynomials in $\mathbb{Z}[x]$ to $\mathbb{Z}_p[x]$. By Theorem 4.1, the greatest common divisor of $f_1(x)$ and $f_2(x)$ can be obtained by performing Gaussian elimination on $\mathbf{K}(f_1(x), f_2(x))$. As long as

$$\phi_p(\mathbf{K}(f_1(x), f_2(x))) = \mathbf{K}(\phi_p(f_1(x)), \phi_p(f_2(x))), \tag{28}$$

then $\operatorname{rank} \mathbf{K}(\phi_p(f_1(x)), \phi_p(f_2(x))) \le \operatorname{rank} \mathbf{K}(f_1(x), f_2(x))$, so that the degree of the GCD computed in $\mathbb{Z}_p[x]$ may be too large. In the case of the standard power basis, the property (28) is guaranteed by ensuring that the leading coefficients of $f_1(x)$ and $f_2(x)$ do not vanish under $\phi_p$. In our case, there is a possibility that $\phi_p(c_{t,t-1}) = 0$ for some $t$, so that even if the leading coefficients of $f_1(x)$ and $f_2(x)$ do not vanish, $\mathbf{K}(\phi_p(f_1(x)), \phi_p(f_2(x))$ computed in $\mathbb{Z}_p$ may not have the correct structure. Thus, we further require that $\phi_p\left(C_{n_2+1}^{n_1+n_2-1}\right) \neq 0$. We summarize this below.

THEOREM 6.1. *Suppose that $p$ does not divide any one of* $\text{lcoeff}(f_1(x))$, $\text{lcoeff}(f_2(x))$, *or* $C_{n_2+1}^{n_1+n_2-1}$. *If* $g(x) = \gcd(f_1(x), f_2(x)) \in \mathbb{Z}[x]$ *and* $g_p(x) = \gcd(\phi_p(f_1(x)), \phi_p(f_2(x))) \in \mathbb{Z}_p[x]$, *then* $\deg g_p(x) \geq \deg g(x)$.

We say that $p$ is *unlucky* if it divides $\text{lcoeff}(f_1(x))$, $\text{lcoeff}(f_2(x))$, or $C_{n_2+1}^{n_1+n_2-1}$, or if $\deg g_p(x) > \deg g(x)$. Notice that a prime $p$ can be lucky by our definition, yet unlucky if the GCD is computed by first converting the polynomials to the standard power basis. For example, the leading coefficients of Hermite polynomials are powers of 2, so that 2 is unlucky if the polynomials are first converted to the standard power basis. However, the same prime $p$ may be lucky if we compute a GCD without converting to the standard power basis because of our choice of the special element $z = 2x$.

For the issue of normalization of the images, let $g(x) = \gcd(f_1(x), f_2(x))$ in $\mathbb{Z}[x]$ with degree $d$. If we assume that $f_1(x)$ and $f_2(x)$ are primitive, then $C_{d+1}^{n_1} \cdot \text{lcoeff}(g(x)) \mid \text{lcoeff}(f_1(x))$ and $C_{d+1}^{n_2} \cdot \text{lcoeff}(g(x)) \mid \text{lcoeff}(f_2(x))$. Thus, $C_{d+1}^{n_1} \cdot \text{lcoeff}(g(x))$ is a common divisor of $\text{lcoeff}(f_1(x))$ and $C_{n_2+1}^{n_1} \cdot \text{lcoeff}(f_2(x))$. Thus, once an image is computed in $\mathbb{Z}_p$, we normalize its leading coefficient to be

$$\gcd\left(\text{lcoeff}(f_1(x)), C_{n_2+1}^{n_1} \cdot \text{lcoeff}(f_2(x))\right) \cdot \left(C_{d+1}^{n_1}\right)^{-1} \bmod p. \tag{29}$$

Note that this requires $\phi_p\left(C_{d+1}^{n_1}\right) \neq 0$. Combined with the conditions needed in Theorem 6.1, it follows that $p$ must not divide $C_1^{n_1+n_2-1}$.

Finally, termination is straightforward since the GCD must be a solution to the linear system of equations specified by the striped Krylov matrix $\mathbf{K}$, so the standard technique of applying Hadamard's inequality lead to a bound on the number of lucky primes needed. One may also use trial division because the degree of the computed GCD cannot be too small. It is therefore clear how to obtain a modular algorithm that is analogous to that of Brown [10].

We also note that since our fraction-free algorithm is based on the FFFG elimination algorithm, we can obtain another modular algorithm that computes the cofactors as well, using techniques similar to those used in [12, 13]. In addition, this algorithm is output-sensitive, so that the number of images needed depends on the size of the output and not on the a priori Hadamard's inequality. Output sensitivity is obtained by examining the associated linear system [11]. We refer the reader to the references for more details.

## 6.1 Conversion Cost

It is of interest to see whether it is better to perform the Euclidean algorithm with polynomial division in alternate basis or in the standard power basis with the added cost of converting the input and the output between the two bases. For simplicity, we assume that $n = n_1 = n_2$. We also assume that the standard Euclidean algorithm is used in both cases for a fair comparison, because our algorithms are direct generalizations of the standard ones. We will limit our attention to the case of bases consisting of orthogonal polynomials, but the other cases are similar.

If we perform the modular algorithm in alternate basis, there is no conversion cost. Thus, we concentrate on the division steps and count the number of operations *in addition* to those needed in a standard polynomial division algorithm. Each division step requires the computation of $z^j \cdot R_i(x)$ for some $j = 1, \ldots, n_{i-1} - n_i$ when dividing by $R_i(x)$. If $R_k(x)$ is the last element of the PRS, then $\deg R_k(x) = n_k = d$. It follows that during the Euclidean algorithm we would need to compute $z \cdot F_i(x)$ for some $F_i(x)$ with $\deg F_i(x) = i$, $i = d, \ldots, n-1$. Now, the computation of $z^j \cdot F_i(x)$ requires $3i + 2$ multiplications and $2(i-1)$ additions in $\mathbb{Z}_p$. Summing over $i = d \ldots, n-1$ gives a total of

$$\begin{aligned} n(3n+1)/2 - d(3d+1)/2 &\quad \text{multiplications} \\ n(n-3) - d(d-3) &\quad \text{additions.} \end{aligned}$$

Note that if $c_{t,t-1} = 1$, then the number of multiplications becomes $n^2 - d^2$.

On the other hand, if we first convert the input polynomials to standard power basis, it takes $(n+1)(n+2)$ multiplications and $n(n+1)$ additions in $\mathbb{Z}_p$. The conversion of the result back to alternate basis requires $d+1$ divisions, $(d+1)(d+4)/2$ multiplications, and $(d+1)(d+2)/2 - 1$ additions in $\mathbb{Z}_p$. Thus, the total conversion cost is

$$\begin{aligned} d+1 &\quad \text{divisions} \\ (n+1)(n+2) + (d+1)(d+4)/2 &\quad \text{multiplications} \\ n(n+1) + (d+1)(d+2)/2 - 1 &\quad \text{additions.} \end{aligned}$$

The effect of division is more difficult to analyze, but from the above we see that if $c_{t,t-1} = 1$ then performing the Euclidean algorithm directly without conversion is certainly more effective with respect to all operations. This is applicable to a number of choices of polynomial basis (see Remark 2.4). In the general case, the number of multiplications in the two approaches depend on the relative size of $n$ and $d$. Generally if $d$ is large then it is better to perform the Euclidean algorithm in the alternate basis, while it is better to perform the conversion if $d$ is small. The crossover point is at $d \approx n/2$. Intuitively, if $d$ is large then the number of steps in the divisions is small, so the number of times one has to compute $z \cdot F_i(x)$ is also small. Finally, we note that in the case of Newton basis the number of multiplications needed in the computation of $z \cdot F_i(x)$ is only $i$, so the total number of additional multiplications required to perform the Euclidean algorithm without conversion is

$$n(n-1)/2 - d(d-1)/2,$$

leading to a more significant improvement.

## 7. CONCLUSIONS

In this paper we have considered the computation of GCDs of polynomials represented in non-standard power bases. The computations are to be done without conversion into the standard power basis and are meant for exact arithmetic domains where coefficient growth is an issue. We have given both fraction-free and modular algorithms for GCD computations.

There are a number of additional extensions for exact computation of polynomials in non-standard bases which we plan to pursue in the future. These include effective computation of GCDs of more than two polynomials and more generally matrix GCDs of matrices of polynomials. In the latter computation the resulting answers are typically required to be in matrix normal form. As such we plan to investigate computation of normal forms for matrices of polynomials in non-standard forms.

The alternate bases that we have given results for do not, unfortunately include polynomials represented in terms of Lagrange basis polynomials. Such a representation would in effect allow for two polynomials to be represented in terms of their values at certain interpolation points with the result being the values of the polynomial GCD at these same values. Such a procedure does not follow from our approach since we make use of the elimination of higher order coefficients. In the Lagrange basis case such elimination does not result in a remainder sequence of reduced degrees. Indeed elimination of higher order terms does not tell us anything about the degree of the polynomial, a difficulty for our approach.

We currently do not take advantage of cases when a polynomial has a sparse representation in a specific basis. It would be interesting to see if our results can be applied to "black box polynomials" as well [15, 20].

## 8. REFERENCES

[1] M. Abramowitz and I. Stegun. *Handbook of Mathematical Functions*. Dover, 1974.

[2] E. Bareiss. Sylvester's identity and multistep integer-preserving Gaussian elimination. *Math. Comp.*, 22(103):565–578, 1968.

[3] S. Barnett. Greatest common divisors of several polynomials. *Proc. Cambridge Phil. Soc.*, 70:263–468, 1971.

[4] S. Barnett. *Polynomial and Linear Control Systems*. Marcel-Dekker, 1983.

[5] S. Barnett. Division of generalized polynomials using the comrade matrix. *Linear Algebra and its Applications*, 60:159–175, 1984.

[6] S. Barnett. Euclidean remainders for generalized polynomials. *Linear Algebra and its Applications*, 99:111–122, 1988.

[7] B. Beckermann and G. Labahn. Effective computation of rational approximants and interpolants. *Reliable Computing*, 6:365–390, 2000.

[8] B. Beckermann and G. Labahn. Fraction-free computation of matrix rational interpolants and matrix GCDs. *SIAM J. Matrix Anal. Appl.*, 22(1):114–144, 2000.

[9] W. Brown and J. Traub. On Euclid's algorithm and the theory of subresultants. *J. ACM*, 18(4):505–514, 1971.

[10] W. S. Brown. On Euclid's algorithm and the computation of polynomial greatest common divisors. *J. ACM*, 18(4):478–504, 1971.

[11] S. Cabay. Exact solution of linear equations. In *Proceedings of the Second Symposium on Symbolic and Algebraic Manipulation*, pages 392–398, 1971.

[12] H. Cheng. *Algorithms for Normal Forms for Matrices of Polynomials and Ore Polynomials*. PhD thesis, University of Waterloo, 2003.

[13] H. Cheng and G. Labahn. Output-sensitive modular algorithms for polynomial matrix normal forms. *Submitted to J. Symbolic Computation*, 2004.

[14] G. E. Collins. Subresultants and reduced polynomial remainder sequences. *J. ACM*, 14(1):128–142, 1967.

[15] A. Díaz and E. Kaltofen. On computing greatest common divisors with polynomials given by black boxes for their evaluation. In *Proc. 1995 Internat. Symp. Symbolic Algebraic Comput. (ISSAC'95)*, pages 232–239, 1995.

[16] G.M. Diaz-Toca and L. Gonzalez-Vega. Square-free decompositioon of univariate polynomials depending on a parameter. application to the inteergration of parametric rational functions. *J. Symbolic Computation*, 32(3):191–209, 2001.

[17] G.M. Diaz-Toca and L. Gonzalez-Vega. Barnett's theorems about the greatest common divisor of several univariate polynomials through bezout-like matrices. *J. Symbolic Computation*, 34(1):59–81, 2002.

[18] K. O. Geddes, S. R. Czapor, and G. Labahn. *Algorithms for computer algebra*. Kluwer Academic Publishers, 1992.

[19] L. Gemignani. Manipulating polynomials in generalized form. 1996.

[20] E. Kaltofen and B. Trager. Computing with polynomials given by black boxes for their evaluations: Greatest common divisors, factorization, separation of numerators and denominators. *J. Symbolic Computation*, 9(3):301–320, 1990.

[21] Z. Li. *A Subresultant Theory for Linear Differential, Linear Difference and Ore Polynomials, with Applications*. PhD thesis, Johannes Kepler University, 1996.

[22] R. Loos. Generalized polynomial remainder sequences. In *Computer Algebra: Symbolic and Algebraic Computation*, pages 115–137. Springer-Verlag, 1982.

[23] B. Mishra. *Algorithmic Algebra*. Springer-Verlag, 1993.

[24] J. von zur Gathen and J Gerhard. *Modern Computer Algebra*. Cambridge University Press, 1999.