On Computing Greatest Common Divisors with Polynomials Given By Black Boxes for Their Evaluations*

Angel Díaz and Erich Kaltofen

Department of Computer Science, Rensselaer Polytechnic Institute
Troy, New York 12180-3590
{diaza,kaltofen}@cs.rpi.edu
http://www.cs.rpi.edu/~kaltofen

Abstract

The black box representation of a multivariate polynomial is a function that takes as input a value for each variable and then produces the value of the polynomial. We revisit the problem of computing the greatest common divisor (GCD) in black box format of several multivariate polynomials that themselves are given by black boxes. To this end an improved version of the algorithm sketched in Kaltofen and Trager [*J. Symbolic Comput.*, vol. 9, nr. 3, p. 311 (1990)] is described. Also the full analysis of the improved algorithm is given. Our algorithm constructs in random polynomial-time a procedure that will evaluate a fixed associate of the GCD at an arbitrary point (supplied as its input) in polynomial time. The randomization of the black box construction is of the Monte-Carlo kind, that is with controllably high probability the procedures evaluating the GCD are correct at all input points. Finally, a Maple prototype implementation as well as our plans for developing a subsystem for manipulating multivariate polynomials and rational functions in black box representation are presented.

1 Introduction

The dreaded phenomenon of expression swell in symbolic computation can be palliated by adopting implicit representations for symbolic objects, such as by straight-line programs (Kaltofen 1988 and 1989) or by so-called black box representations (Kaltofen and Trager 1990). In the latter, each expression is a symbolic object, more specifically, a computer program with a set of statically initialized data, which takes as input a value for each variable and then produces the value of the symbolic object it represents at the specified point. Efficient algorithms for factoring multivariate polynomials in the black box representation are described in Kaltofen and Trager (1990). Various efficient methods for recovering the sparse representation of a polynomial or rational function, for example, as lists of non-zero monomials, have also been previously discovered (Zippel 1979 and 1990, Ben-Or and Tiwari 1988, Grigoriev et al. 1990, Kaltofen and Lakshman 1988, Kaltofen et al. 1990, Mansour 1992, Grigoriev and Karpinski 1993, Lakshman and Saunders 1994 and 1995, Grigoriev and Lakshman 1994).

In this article, we revisit the problem of computing the black box greatest common divisor (GCD) of several multivariate polynomials that are given by black boxes. In Kaltofen and Trager (1990, p. 311), we have presented a brief sketch of a possible solution. Here we give both the detailed description and a full analysis of an improved version of that algorithm. We also discuss the prototypical implementation of our algorithm in Maple and the issues that arise for building a system for manipulating polynomials in black box representation that is callable from general purpose computer algebra software such as Axiom, Maple, or Mathematica.

The black box GCD algorithm must fix a unique scalar multiple of the GCD that the produced black box will evaluate at the given points. The scalar multiple is fixed by performing a generic symbolic shift of the variables that makes the shifted GCD monic in the main variable X. By our shifts, we also avoid the computationally costly content and primitive part problem of GCD algorithms for polynomials in explicit representation like Brown's (1971), Zippel's (1979), or Char's et al. (1989) modular algorithms, and like Moses's and Yun's (1973), Wang's (1980), or Kaltofen's (1985) Hensel-lifting based algorithms. Note that the shifts do not introduce expression swell since the representation of the input polynomials is by black boxes. Values of the GCD are obtained by introducing a second variable Y in the fashion of homotopy continuation (Drexler 1977). For $Y = 0$, the black box computes the value of the GCD at a predetermined point, while for $Y = 1$ we will have the value of the GCD at the inputs to the black box. Finally, the problem of computing the GCD of several polynomials is reduced to the problem of computing the GCD of a pair of polynomials by taking a random scalar sum, which is a technique first introduce by D. Spear (see Wang 1980, p. 57). Finally, we remark that our algorithm uses randomization and that the resulting black box for the GCD may with controllably small probability be incorrect. However, if the black box was determined correctly, the values for the GCD

*This material is based on work supported in part by the National Science Foundation under Grant No. CCR-9319776 and by GTE under a Graduate Computer Science Fellowship (second author).

will always be correct.

Our improvement of the algorithm sketched in Kaltofen and Trager (1990) is as follows. By using a modular GCD approach for the arising bivariate GCD problem (in the variables X and Y) we can greatly reduce the number of calls to the black boxes for the input polynomials. In fact, under fortunate circumstances the input black boxes need only be called degree of the input polynomials many times in order to obtain the value of the GCD at a single point.

The black box approach to computing GCDs of multivariate polynomials has several advantages over other methods. First, it can handle polynomials that are exponential in size even when represented in sparse format, such as determinants of matrices with polynomial entries. Second, the produced GCD can be manipulated in various ways. For instance, the new sparse interpolation algorithms with respect to non-standard bases can retrieve a concise representation even if the GCD is not sparse in the power basis sense (Lakshman and Saunders 1994 and 1995, Grigoriev and Lakshman 1994). An example would be a GCD that is sparse by shifting the variables, like $(x_1-1)\cdot(x_2-2)\cdots(x_n-n)+1$. Third, the black box representation improves on the straight-line program representation (Kaltofen 1988) in that the produced objects, in our case the black box programs for the GCD, are very small in size. In fact, the black box representation was invented because of the experience obtained from experiments in our DAGWOOD system for manipulating straight-line programs (Freeman et al. 1988).

Most sparse interpolation algorithms are ideally suited for parallelization: the algorithms probe the polynomial at selected points and then perform the interpolation task by use of the obtained values. Therefore, the evaluation at the different points can be done on different computers. Rayes et al. (1994) have implemented the sparse GCD algorithm by Zippel (1979) in that way. However, their approach is restricted to a single sparse interpolation method (Zippel's), and their algorithm still faces the content and leading coefficient problems. Furthermore, due to the small size of our resulting black box programs for the GCD, sparse interpolation can be parallelized on a network of computers with less data transfer (cf. Díaz et al. 1991) than one would have if the input polynomials were represented in standard format or if the resulting GCD were represented as a straight-line program.

Our paper is organized as follows. In §2, we describe the black box GCD algorithm in full detail and state its computational complexity. We give running time measures for both the task of constructing the black box and for an evaluation of the produced black box at a point. In §3, we supply the proofs of our analysis including the estimate on the probability that the algorithm fails to produce a correct result. In §4, we describe our Maple prototype implementation and concomitantly lay out our plans of an implementation in a compilable language.

2 The Black Box GCD Algorithm

We now give a detailed description of our algorithm. For clarity, we shall state the actual computations in imperative mood and typeset them in italics font, while keeping the extensive comments in narrative and roman.

Algorithm *Black Box GCD*

Input: A black box for each polynomial $f_i(x_1,\ldots,x_n) \in \mathsf{K}[x_1,\ldots,x_n]$ for $i=1,\ldots,r$, $r\geq 2$, where K is a field.

Output: A program (see Figure 1) that makes calls to the black boxes of the f_i's and has with probability no less than $1-\epsilon$ the following property. The program accepts n field elements $p_1,\ldots,p_n$. It returns $g(p_1,\ldots,p_n)\in\mathsf{K}$ where $g=\mathrm{GCD}_{1\leq i\leq r}(f_i)$. Notice that $g(p_1,\ldots,p_n)$ is determined only up to a multiple in K. For repeated invocations with different arguments, it returns the value scaled by the same multiple. Notice also that the failure probability applies to the construction and not to the execution of the program. That is, with probability at least $1-\epsilon$ the output program is correct; a correct program will always produce the true values of the GCD.

Step 1: *Pick random field elements*

$$a_2,\ldots,a_n,b_2,\ldots,b_n,c_3,\ldots,c_r$$

from a sufficiently large finite subset $\mathsf{R}\subset\mathsf{K}$. We will give the cardinality of this set in relation to $\deg(f_i)$ for $1\leq i\leq r$ and ϵ in the statement of theorem 1 below.

Let the overline operator $\bar{}$ for any $h\in\mathsf{K}[x_1,\ldots,x_n]$ be the projection

$$\bar{h}(X,Y)=h(X,Y(p_2-a_2p_1-b_2)+a_2X+b_2,\\ \ldots,Y(p_n-a_np_1-b_n)+a_nX+b_n).$$

Note that $\bar{h}(p_1,1)=h(p_1,\ldots,p_n)$. We will be using the bivariate polynomials

$$\bar{f}_0(X,Y)=\bar{f}_2(X,Y)+\sum_{i=3}^{r}c_i\bar{f}_i(X,Y),\ \bar{f}_1(X,Y),$$
$$\gamma(X,Y)=\mathrm{GCD}(\bar{f}_0(X,Y),\bar{f}_1(X,Y)),$$

and the univariate GCDs

$$\bar{\gamma}_e(X)=\mathrm{GCD}(\bar{f}_0(X,e),\bar{f}_1(X,e))\quad\text{for}\quad e\in\mathsf{K}.$$

Among the possible associates for the GCDs γ and $\bar{\gamma}_e$ we always choose those whose leading coefficient is one an the same element of K, namely the leading coefficient of $\bar{\gamma}_0$. We will show in the proof of Theorem 1 below that with high probability $\gamma=\bar{g}$.

Step 2: *By standard interpolation compute*

$$\bar{f}_0(X,0)=f_2(X,a_2X+b_2,\ldots,a_nX+b_n)+\\ \sum_{i=3}^{r}c_if_i(X,a_2X+b_2,\ldots,a_nX+b_n)$$

and

$$\bar{f}_1(X,0)=f_1(X,a_2X+b_2,\ldots,a_nX+b_n);$$

then compute $\bar{\gamma}_0(X)=\mathrm{GCD}(\bar{f}_0(X,0),\bar{f}_1(X,0))$.

The interpolation algorithms mentioned above and below need to know $\deg_X(\bar{f}_i)$ for all $1\leq i\leq r$. Either the degree or an upper bound is supplied as input, or the degree can be probabilistically guessed as follows (see Kaltofen and Trager 1990). Pick a random $B\in\mathsf{R}$ and compute $\bar{f}_i(X,B)$ by determining a succession of polynomials $\bar{f}_i^{(d)}(X,B)$ for $d=1,2,3,\ldots$ until $\bar{f}_i^{(d)}(X,B)=\bar{f}_i(X,B)$, where $\bar{f}_i^{(d)}(X,B)$ is the interpolate at $X=0,1,\ldots,d$ of $\bar{f}_i(X,B)$. We test whether $\bar{f}_i^{(d)}(X,B)=\bar{f}_i(X,B)$ by evaluating at a random $A\in\mathsf{R}$: if $\bar{f}_i^{(d)}(A,B)=\bar{f}_i(A,B)$ then

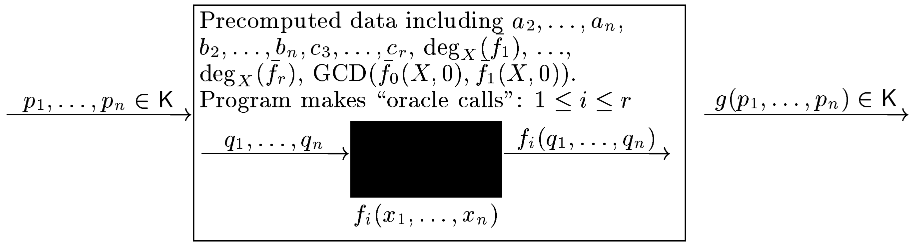

Figure 1: The program for evaluating the GCD of black box

we declare $\bar{f}_i^{(d)}(X, B) = \bar{f}_i(X, B)$ and $\deg_X(\bar{f}_i) = d$, otherwise we continue the interpolation. The failure probability is bounded by $\deg(\bar{f}_i)/\mathrm{card}(\mathsf{R})$, which means that the degree in X of $\bar{f}_i(X, Y)$ would have to be very large in order for the guessing procedure to be unreliable. Note that one may use $B = 0$ provided the leading coefficient of $\bar{f}_i(X, Y)$ does not depend on Y. One can estimate the probability that the random a_i yield this condition, but we will not incorporate such an estimate in our analysis. Ultimately, if the probability of determining the degree correctly is to be guaranteed, an upper bound for all $\deg(f_i)$ must be provided by the user of this algorithm.

If not all polynomials f_i for $i \ge 3$ are needed to determine $\bar{\gamma}_0$, those polynomials that are redundant meaning that they can be omitted without affecting the GCD, preferably the ones of highest degree, can be ignored for future considerations. An optimization at this point can significantly reduce the complexity of the output black box.

Step 3: This step constructs the programs for evaluation of g at $p_1, \ldots, p_n$ as described in the output specifications. First $a_2, \ldots, a_n, b_2, \ldots, b_n, c_3, \ldots, c_r$, $d_i = \deg_X(\bar{f}_i)$ for $1 \le i \le r$, $\bar{\gamma}_0$, and $\delta = \deg(\bar{\gamma}_0)$ are "hardwired" into that program. Then the following Steps A, and B are appended to the program.

Step A: Let $\mathsf{S} \subset \mathsf{K}$ containing $\{1\}$ be of cardinality at least $d_0 d_1 + \delta$. *First select* $i_1 = 1$ *and compute* $\bar{\gamma}_1(X)$. *If* $\deg(\bar{\gamma}_1(X)) = \delta$ *then* the black box can terminate early, therefore *return* $\bar{\gamma}_1(p_1)$ *as the requested evaluation. If* $\deg(\bar{\gamma}_1(X)) > \delta$ *compute* γ *by using a modular* GCD *approach. To this end select* $i_1, i_2, \ldots, i_\delta$ *pairwise distinct elements in* S *such that* $\deg(\bar{\gamma}_{i_j}) = \delta$ *for* $0 \le j \le \delta$. Any element e in S such that $\deg(\bar{\gamma}_e) > \delta$ must be discarded. Further we are guaranteed that only $d_0 d_1$ of such elements exist. If for any element e in S we have $\deg(\bar{\gamma}_e) < \delta$ then the black box is invalid and the program returns an error indicating that a new black box is needed with different random elements. The polynomials $\bar{\gamma}_e$ are computed as GCDs of $\bar{f}_0(X, e)$ and $\bar{f}_1(X, e)$ which can be computed by interpolation using d_i as degree bound for each $\bar{f}_i(X, e)$.

Step B: Let $I = \{i_0 = 0, i_1, i_2, \ldots, i_\delta\}$, let $P = \{\bar{\gamma}_{i_0}(X), \bar{\gamma}_{i_1}(X), \ldots, \bar{\gamma}_{i_\delta}(X)\}$ with each polynomial having its leading coefficient $\mathrm{ldcf}(\bar{\gamma}_0(X))$, and let COEFF$(i, P)$ for $0 \le i \le \delta$ be a function that returns an ordered coefficient list corresponding to the i^{th} coefficient of X^i of all the polynomials in P. Furthermore, let INTER(I, C, V) be a function that computes the polynomial in the variable V which interpolates the points in I at the values given by C.

$\gamma \leftarrow X^\delta$;
For $j \leftarrow 0, \ldots, \delta - 1$ *Do:*
Let $\gamma = \gamma + X^j \cdot \mathrm{INTER}(I, \mathrm{COEFF}(i, P), Y)$;

Finally $\gamma(p_1, 1)$ *can now be returned at the requested evaluation.* □

The following theorem states the complexity and the failure estimate in relation to the cardinality of R mentioned in Step 1 of the above algorithm.

Theorem 1 *The Black Box Polynomial GCD algorithm can construct its output program in polynomially many arithmetic steps as a function of* r *and* $\deg(f_i)$ *for* $1 \le i \le r$. *For each* i, *it requires* $\deg(f_i) + 1$ *oracle calls to the black box of the polynomial if the degree of each polynomial is known, or* $\deg(f_i) + 2$ *oracle calls to the black box of the polynomial if only a degree bound is known. If the cardinality of the set* R *in Step 1 is chosen*

$$\mathrm{card}(\mathsf{R}) \ge \deg(f_1) \cdot (1 + 2 \max_{2 \le i \le r} \{\deg(f_i)\}) \Big/ \epsilon$$

then the algorithm succeeds with probability no less than $1 - \epsilon$ *and the resulting program will always correctly evaluate the GCD at all points. That program in turn can be executed in polynomially many arithmetic steps and at best with* $\deg(f_i) + 1$ *many oracle calls to each black box of* f_i *for all* $1 \le i \le r$, *and at worst with*

$$\deg(f_i) \cdot (\delta + \deg(f_1) \cdot \max_{2 \le i \le r} \{\deg(f_i)\})$$

many oracle calls to each black box of f_i, *where* $\delta = \deg(\mathrm{GCD}_{1 \le i \le r}(f_i))$.

Note that our analysis supposes that the degrees of the input polynomials (or upper bounds for them) are known. As explained in Step 2, it is possible to probabilistically determine them. The proof of the above theorem is somewhat involved and is given in the next section. Those readers interested in the implementation of our algorithm may skip to the subsequent section.

3 Complexity Analysis

In this section we provide a proof for the complexity and probabilistic analysis of the Black Box GCD algorithm as given in Theorem 1. We will base our arguments on three lemmas, which we state and prove next. First, we shall establish a well-known condition under which GCD operations and homomorphic imaging commute. In order to avoid ambiguities, we sometimes write $\mathrm{GCD}_{\mathsf{D}}(f_1, f_2)$ meaning that the GCD is to be taken in the domain D. Note that our condition is weaker than those established by Brown (1971), who assumes that no leading coefficient of the inputs an no leading coefficient of a non-zero remainder in the Euclidean chain is mapped to zero. Consequently, we will obtain better probabilistic estimates.

Lemma 1 *Let* E *be a* UFD, K *a field,* $\phi: \mathsf{E} \to \mathsf{K}$ *be a ring homomorphism,* $f_1 = a_n x^n + \cdots + a_0$ *and* $f_2 = b_m x^m + \cdots + b_0 \in \mathsf{E}[x]$, $l = \mathrm{ldcf}_x(\mathrm{S}_\delta(f_1, f_2))$ *where* $\delta = \deg(\mathrm{GCD}_{\mathsf{E}[x]}(f_1, f_2))$ *and where* $\mathrm{S}_\delta(f_1, f_2)$ *is the subresultant formal degree* δ *of* f_1 *and* f_2 *(see Brown and Traub 1971). If* $\phi(l) \neq 0$ *we have*

$$u \cdot \phi(\mathrm{GCD}_{\mathsf{E}[x]}(f_1, f_2)) = \mathrm{GCD}_{\mathsf{K}[x]}(\phi(f_1), \phi(f_2))$$

and

$$\delta = \deg(\mathrm{GCD}_{\mathsf{K}[x]}(\phi(f_1), \phi(f_2))) \tag{1}$$

for $u \in \mathsf{K} \setminus \{0\}$.

Proof From the condition $\phi(l) \neq 0$ we can infer that one or both of the $\mathrm{ldcf}_x(\phi(f_1))$ and $\mathrm{ldcf}_x(\phi(f_2))$ must be nonzero. Let $c_\delta = \mathrm{ldcf}_x(\mathrm{GCD}_{\mathsf{E}[x]}(f_1, f_2))$. Our first claim is that

$$\delta = \deg(\phi(\mathrm{GCD}_{\mathsf{E}[x]}(f_1, f_2))),$$

where δ was defined as $\deg(\mathrm{GCD}_{\mathsf{E}[x]}(f_1, f_2))$. We observe without loss of generality that if $\phi(a_n) \neq 0$, $c_\delta \mid a_n$ implies that $\phi(c_\delta) \neq 0$, which proves the claim. Let

$$\delta' = \deg(\mathrm{GCD}_{\mathsf{K}[x]}(\phi(f_1), \phi(f_2))).$$

Since ϕ is a ring homomorphism

$$\phi(\mathrm{GCD}_{\mathsf{E}[x]}(f_1, f_2)) \mid \mathrm{GCD}_{\mathsf{K}[x]}(\phi(f_1), \phi(f_2)) \tag{2}$$

and consequently,

$$\delta \leq \delta'. \tag{3}$$

Let $f_1, f_2, \ldots, f_k$ be the polynomial remainder sequence (PRS) of f_1 and f_2. Also, let $\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_{\hat{k}}$ be the PRS of $\phi(f_1)$ and $\phi(f_2)$. Assume for purpose of contradiction that $\delta' > \delta$. According to the Fundamental Theorem of Subresultants $\mathrm{S}_j(\hat{f}_1, \hat{f}_2) = 0$ for $j = 0, \ldots, \delta' - 1$. Since δ is in the range $0, \ldots, \delta' - 1$, $\mathrm{S}_\delta(\hat{f}_1, \hat{f}_2)) = 0$.

Case 1: Both $\mathrm{ldcf}_x(\phi(f_1))$ and $\mathrm{ldcf}_x(\phi(f_2))$ are nonzero :

The dimension of the matrix corresponding to $\phi(\mathrm{S}_\delta(f_1, f_2))$ is $(n + m - 2\delta) \times (n + m - 2\delta)$. Since $\mathrm{ldcf}_x(\phi(f_1))$ and $\mathrm{ldcf}_x(\phi(f_2))$ are nonzero the degree of $\phi(f_1)$ and $\phi(f_2)$ does not change, and therefore the dimension of the matrix corresponding to $\mathrm{S}_\delta(\phi(f_1), \phi(f_2))$ is also $(n+m-2\delta) \times (n+m-2\delta)$. From this it follows that when $\mathrm{ldcf}_x(\phi(f_1))$ and $\mathrm{ldcf}_x(\phi(f_2))$ are nonzero $\phi(\mathrm{S}_\delta(f_1, f_2)) = \mathrm{S}_\delta(\phi(f_1), \phi(f_2))$ and hence a contradiction.

Case 2: Without loss of generality, $\mathrm{ldcf}_x(\phi(f_1)) = 0$ and $\mathrm{ldcf}_x(\phi(f_2)) \neq 0$:

Consider the following example.

$$\phi(\mathrm{S}_\delta(f_1, f_2)) = \phi(\mathrm{Det}(\begin{bmatrix} a_n & a_{n-1} & a_{n-2} & \cdots \\ 0 & a_n & a_{n-1} & \cdots \\ & \ddots & & \ddots \\ b_m & b_{m-1} & b_{m-2} & \cdots \\ 0 & b_m & b_{m-1} & \cdots \\ & \ddots & & \ddots \end{bmatrix}))$$

$$= \mathrm{Det}(\begin{bmatrix} 0 & \phi(a_{n-1}) & \phi(a_{n-2}) & \cdots \\ 0 & 0 & \phi(a_{n-1}) & \cdots \\ & \ddots & & \ddots \\ \phi(b_m) & \phi(b_{m-1}) & \phi(b_{m-2}) & \cdots \\ 0 & \phi(b_m) & \phi(b_{m-1}) & \cdots \\ & \ddots & & \ddots \end{bmatrix})$$

and

$$\mathrm{S}_\delta(\phi(f_1), \phi(f_2)) = \mathrm{Det}(\begin{bmatrix} \phi(a_{n-1}) & \phi(a_{n-2}) & & \cdots \\ 0 & \phi(a_{n-1}) & \phi(a_{n-2}) & \cdots \\ & \ddots & & \ddots \\ \phi(b_m) & \phi(b_{m-1}) & \phi(b_{m-2}) & \cdots \\ 0 & \phi(b_m) & \phi(b_{m-1}) & \cdots \\ & \ddots & & \ddots \end{bmatrix}).$$

The dimension of $\phi(\mathrm{S}_\delta(f_1, f_2))$ is again $(n+m-2\delta) \times (n+m-2\delta)$. However, since $\mathrm{ldcf}_x(\phi(f_1)) = 0$, there is a degree drop (in the above example by 1) in $\phi(f_1)$ and hence the dimension of $\mathrm{S}_\delta(\phi(f_1), \phi(f_2))$ is now $(n+m-2\delta-1) \times (n+m-2\delta-1)$. By minor expansion of the first column of $\phi(\mathrm{S}_\delta(f_1, f_2))$, we see that $\phi(\mathrm{S}_\delta(f_1, f_2)) = \mathrm{ldcf}_x(\phi(f_2)) \cdot \mathrm{S}_\delta(\phi(f_1), \phi(f_2))$. In the case more coefficients of $\phi(f_1)$ map to zero we see that $\phi(\mathrm{S}_\delta(f_1, f_2)) = \pm\, \mathrm{ldcf}_x(\phi(f_2))^\alpha \cdot \mathrm{S}_\delta(\phi(f_1), \phi(f_2))$, where α is the number of coefficients of $\phi(f_1)$ mapping to zero before the first nonzero coefficient is reached. We can be guaranteed that both $\phi(f_1)$ and $\phi(f_2)$ can not vanish identically since $\phi(l) \neq 0$. This is again a contradiction to the assumption.

From the contradiction between what was arrived at from the Fundamental Theorem of Subresultants and Case 1 and 2, we know that $\delta' \leq \delta$, hence,

$$\delta \geq \delta'. \tag{4}$$

From (3) and (4)

$$\delta = \delta'. \tag{5}$$

Finally (1) follows from (2) and (5). □

The next lemma concerns the reduction of the GCD problem of many polynomials to computing the GCD of a pair of polynomials. Here we follow a strategy first used by Spear and extended to multivariate polynomials by Kaltofen (1988, Theorem 6.2). Note that the usage of only $r - 2$ random elements can also be found in von zur Gathen et al.(1994).

Lemma 2 *Let* $f_i(x_1, \ldots, x_n) \in \mathsf{K}[x_1, \ldots, x_n]$ *be nonzero polynomials for* $i = 1, \ldots, r$, $r \geq 2$, K *a field,* $d = \deg(f_1)$ *for* $1 \leq i \leq r$, $\mathsf{R} \subset \mathsf{K}$. *Then for randomly chosen* $c_i \in \mathsf{R}$, $3 \leq i \leq r$ *we have,*

$$\Pr(\mathrm{GCD}_{1 \leq i \leq r}(f_i) = \mathrm{GCD}(f_1, f_2 + \sum_{i=3}^{r} c_i f_i)) \geq 1 - d \big/ \mathrm{card}(\mathsf{R})$$

Proof We first show this lemma for $n = 1$. Let

$$\hat{f}_1 = f_1, \quad \hat{f}_2 = f_2 + \sum_{i=3}^{r} \gamma_i f_i \in \mathsf{E}[x], \qquad \mathsf{E} = \mathsf{K}[\gamma_3 \ldots \gamma_r],$$

$\gamma_3 \ldots \gamma_r$ be indeterminants, and let $g = \mathrm{GCD}_{1 \leq i \leq r}(f_i)$. Clearly, $g \mid \hat{f}_1$, $g \mid \hat{f}_2$. The first claim is that $g = \hat{g}$ where $\hat{g} = \mathrm{GCD}(\hat{f}_1, \hat{f}_2)$. We observe that $\hat{g} \in \mathsf{K}[x]$, since $\hat{g}$ divides f_1. Now write $\hat{f}_2 = \hat{g}\hat{f}_2^*$, where $\hat{f}_2^* \in \mathsf{E}[x]$. We know that $\hat{g} \mid f_1$, if we evaluate $\hat{f}_2 = \hat{g}\hat{f}_2^*$ at $\gamma_i = 0$ for $3 \leq i \leq r$, then we see that $\hat{g} \mid f_2$. Evaluating this equation at $\gamma_i = 1$ and $\gamma_j = 0$, $i \neq j$ for $3 \leq i \leq r$ we get $\hat{g} \mid f_2 + f_i$ for $3 \leq i \leq r$ since we know that $\hat{g} \mid f_2$, hence $\hat{g} \mid f_i$ for $3 \leq i \leq r$. Therefore $\hat{g} \mid g$. Consequently since $g \mid \hat{f}_1$ and $g \mid \hat{f}_2$ we know that

$g \mid \hat{g}$, hence we can conclude that $g = \hat{g}$ which proves the first claim. Now let $\phi_{c_3,\ldots,c_r}\colon \mathsf{E} \to \mathsf{K}$ be the ring homomorphism $\gamma_i = c_i$ for $3 \le i \le r$, $l \in \mathsf{E}[x]$ be the $\mathrm{ldcf}_x(\mathrm{S}_\delta(\hat{f}_1, \hat{f}_2))$ where $\delta = \deg(\mathrm{GCD}_{\mathsf{E}[x]}(\hat{f}_1, \hat{f}_2))$. By lemma 1 if for randomly chosen $c_i \in \mathsf{R}$ for $3 \le i \le r$ we have $\phi_{c_3,\ldots,c_r}(l) \ne 0$, then

$$\mathrm{GCD}_{\mathsf{K}[x]}(\phi_{c_3,\ldots,c_r}(\hat{f}_1), \phi_{c_3,\ldots,c_r}(\hat{f}_2)) = \phi_{c_3,\ldots,c_r}(\mathrm{GCD}_{\mathsf{E}[x]}(\hat{f}_1, \hat{f}_2)),$$

which implies the asserted event. Since the $\deg(l) \le d$, the Schwartz (1980)/Zippel (1979) Lemma then establishes the stated probability. The multivariate case can now be reduced to the univariate case as in Theorem 6.2 in Kaltofen (1988). □

By use of Lemma 1 we can now justify the evaluations used in Step 1 of the Black Box GCD algorithm. Because of Lemma 2 we can restrict ourselves to the case of two polynomials.

Lemma 3 *Let $\hat{f}_1$ and $\hat{f}_2$ be nonzero polynomials $\in \mathsf{K}[x_1,\ldots,x_n]$, $d_1 = \deg(\hat{f}_1)$, $d_2 = \deg(\hat{f}_2)$, $\mathsf{R} \subset \mathsf{K}$, $a_2,\ldots,a_n$, $b_2,\ldots,b_n$ be randomly chosen elements $\in \mathsf{R}$, $\phi\colon \mathsf{K}[x_1,\ldots,x_n] \to \mathsf{K}[X]$ be the ring homomorphism generated by $x_1 = X$, $x_i = a_iX + b_i$ for $2 \le i \le n$, $g_1 = \mathrm{GCD}(\hat{f}_1, \hat{f}_2)$ and let $g_2 = \mathrm{GCD}(\phi(\hat{f}_1), \phi(\hat{f}_2))$. Then we have*

$$\Pr\left(\phi(g_1) = g_2 \quad \text{and} \quad \deg(g_1) = \deg(g_2)\right) \ge 1 - 2d_1d_2\Big/\mathrm{card}(\mathsf{R})$$

Proof Let $\psi\colon \mathsf{K}[\alpha_2,\ldots,\alpha_n][x_1,\ldots,x_n] \to \mathsf{K}[\alpha_2,\ldots,\alpha_n, \beta_2, \ldots,\beta_n][X]$ be the ring isomorphism generated by substituting $x_1 = X$, $x_i = \alpha_iX + \beta_i$ for $2 \le i \le n$, let $\phi'\colon \mathsf{K}[\alpha_2,\ldots,\alpha_n,\beta_2,\ldots,\beta_n] \to \mathsf{K}$ be the ring homomorphism generated by $\alpha_i = a_i$, and $\beta_i = b_i$ for $2 \le i \le n$, and let $\tilde{g}_1 = \mathrm{GCD}(\psi(\hat{f}_1)\ \psi(\hat{f}_2))$. We have $\phi = \phi'\psi$ as seen in the diagram in Figure 2.

Due to the nature of the mapping in the ring isomorphism ψ we can see that $\deg(g_1) = \deg_X(\tilde{g}_1)$ since the α_i do not allow the vanishing of $\mathrm{ldcf}_X(\tilde{g}_1)$. Let $l = \mathrm{ldcf}_X(\mathrm{S}_\delta(\psi(\hat{f}_1), \psi(\hat{f}_2)))$, where $\delta = \deg(g_1)$ and S_δ is the δ^{th} subresultant with respect to X. We know that $l \in \mathsf{K}[\alpha_2,\ldots,\alpha_n,\beta_2,\ldots,\beta_n]$ and from the Schwartz/Zippel Lemma

$$\Pr\left(\phi'(l) \ne 0\right) \ge 1 - \deg(l)\Big/\mathrm{card}(\mathsf{R}).$$

The degree of the coefficients with respect to X in $\psi(\hat{f}_1)$ and $\psi(\hat{f}_2)$ can be bounded from above by d_1 and d_2 respectively. There are $d_2 - \delta$ rows of entries in the matrix corresponding to $\mathrm{S}_\delta(\psi(\hat{f}_1), \psi(\hat{f}_2))$ of degree at most d_1 and there are $d_1 - \delta$ rows of entries of degree at most d_2. In the worst case of $\delta = 0$ we can bound the degree of all of the coefficients in above mentioned subresultant by $2d_1d_2$ and hence l. Using Lemma 1 with ϕ', $\psi(\hat{f}_1)$, $\psi(\hat{f}_2)$, $\mathsf{E} = \mathsf{K}[\alpha_2,\ldots,\alpha_n, \beta_2,\ldots,\beta_n]$ and assuming that $\phi'(l) \ne 0$ we can apply Lemma 1, which yields $\phi'(\tilde{g}_1) = g_2$ and $\deg(\tilde{g}_1) = \deg_X(g_2)$. □

Finally, we can prove Theorem 1.

Proof of Theorem 1 The statements on the run time and required black box oracle calls of the algorithm and the returned program are easily verified. First we need to interpolate the univariate polynomials defined in Step 1, namely, $\bar{f}_0(X,0)$ of degree at most $\max_{2\le i\le r}\{\deg(f_i)\}$ and $\bar{f}_1(X,0)$ of degree $\deg(f_1)$. There are $\deg(f_i)+1$ many oracle calls to each individual black box of f_i if the degree is known. If the polynomial is probabilistically guessed as described in Step 2, an extra oracle call for the check at $X = A$ is required. In either case the interpolation and single GCD needed to compute $\gamma_0(X) = \mathrm{GCD}(\bar{f}_0(X,0), \bar{f}_1(X,0))$ described in the algorithm can be accomplished in polynomial time.

The dominating work of the output program is Step A, the computation of GCDs of interpolated univariate polynomials. Let $d_0 = \deg(\bar{f}_0(X,0))$, $d_1 = \deg(\bar{f}_1(X,0))$, and let $\mathsf{S} \subset \mathsf{K}$ containing $\{1\}$ be of cardinality at least $d_0d_1 + \delta$.

As described, if $\deg(\bar{\gamma}_1) = \deg(\bar{\gamma}_0)$ the black box program for the GCD can terminate early only using $r + \sum_{i=1}^{r}\deg(f_i)$ many oracle calls. Both the interpolation and computation of $\bar{\gamma}_1(X)$ can be accomplished in polynomial time. The degree of $\gamma_0(X)$ is, with high probability (see Lemma 3), equal to δ. Then, if $\deg(\bar{\gamma}_1) = \deg(\bar{\gamma}_0)$ the polynomial $\bar{\gamma}_1(X)$ is essentially the GCD described in Lemma 2 and is equal to the homomorphic image of $\mathrm{GCD}_{1\le i\le r}(f_i)$.

At worst $d_0d_1 + \delta$ elements from S are needed to compute $\bar{g}$. The GCD algorithm needs δ "lucky" values for Y to interpolate $\bar{g}$, since the coefficients for $Y = 0$ are already available, and because there can be a maximum of d_0d_1 "unlucky" values, as we will argue below. There are $\deg(f_i)+1$ many oracle calls for each black box of f_i needed to interpolate $\bar{f}_0(X,e)$ and $\bar{f}_1(X,e)$, that for every e in S. Hence, since only $d_0d_1 + \delta$ values from S must be used at the worst, the number of required black box oracle calls follows. Again both the interpolations of $\bar{f}_0(X,e)$ and $\bar{f}_1(X,e)$ and GCD computations can be accomplished in polynomial time. The number of "unlucky" values of $Y = e$ is derived as follows. Let $l(Y) = \mathrm{ldcf}_X(\mathrm{S}_\delta(\bar{f}_0(X,Y), \bar{f}_1(X,Y)))$. The degree of l can be bounded by d_0d_1 and hence only d_0d_1 values can zero the leading coefficient of the subresultant, thus making the computed GCD invalid (cf. Lemma 1).

All that remains is to analyze the failure probabilities. From Lemma 2 we obtain that with probability at least $1 - \deg(f_1)/\mathrm{card}(\mathsf{R})$ we have

$$g = \mathrm{GCD}(f_1,\ldots,f_r) = \mathrm{GCD}(f_1, f_2 + c_3f_3 + \cdots + c_rf_r),$$

when the elements $c_3,\ldots,c_r$ are chosen at random from the set R. Assuming that this is the case, Lemma 3 then yields that with probability at least $1 - 2\deg(f_0)\deg(f_1)/\mathrm{card}(\mathsf{R})$ we have $\mathrm{GCD}(\bar{f}_0(X,0), \bar{f}_1(X,0)) = \bar{g}(X,0)$, where the barred polynomials are defined in Step 1. Furthermore, the leading coefficient of $\bar{g}(X,Y)$ is independent of Y, since by Lemma 3 we also have $\deg(g) = \deg(\bar{g}(X,0))$. Since by virtue of homomorphic imaging we have that $\bar{\gamma}_0$ is a polynomial multiple of $\gamma(X,0)$, these conditions imply that $\gamma(X,Y) = \mathrm{GCD}(\bar{f}_0(X,Y), \bar{f}_1(X,Y)) = \bar{g}(X,Y)$ and that the polynomial γ as determined in Step B is the image of one and the same associate of g, namely the one whose leading coefficient in X has been preselected. Both events occur with probability no less than the product of the stated bounds, which yields the given estimate. □

4 Maple Prototype and Future Plans

By taking advantage of the functionality of the "in house" standard library functions offered by most general purpose computer algebra systems, we were able to prototype our

$$
\begin{array}{ccc}
 & \psi & \\
\mathsf{K}[\alpha_2,\ldots,\alpha_n][x_1,\ldots,x_n] & \rightarrow & \mathsf{K}[\alpha_2,\ldots,\alpha_n,\beta_2,\ldots,\beta_n][X] \\
 & \phi \searrow & \downarrow \phi' \\
 & & \mathsf{K}[X]
\end{array}
$$

Figure 2: Diagram of ψ, ϕ and ϕ'

improved algorithm for computing black box greatest common divisors using Maple.

Our first task was to represent polynomials as a symbolic object which takes as input a value for each variable and then produces the value of the polynomial at the specified point. Maple offers the function `procmake` which takes the "neutral form" of a procedure body and creates an executable procedure. By using the "neutral forms" for statements, local variables, parameters, and several functions we were able to write the Maple procedure `bbpolygen` that when applied to a polynomial returns a procedure that evaluates the polynomial at a given point. As a further exercise we wrote the Maple procedure `bbdmgen` that when given a matrix and the indeterminates of the matrix as input returns a procedure that at a given point first substitutes values for the indeterminates in the matrix and then produces the value of the determinant of the modified matrix. As an example, the Maple code in Figure 4 creates two black boxes that at a given point produces the value of the determinant of their respective Vandermonde matrix.

Finally the Maple procedure `bbgcdgen` which takes as input a list of black boxes, a list of indeterminates, a procedure to generate the random field elements needed in the algorithm description, and a procedure to generate the random field elements needed for degree guessing in the interpolation, returns a program that evaluates correctly the GCD of the black boxes at any point with a probability controlled by the random number generating procedures. The Maple code in Figure 3 shows the construction of a black box that when applied to a point returns the value of the GCD of those black boxes which are computed by code shown in Figure 4.

```
O3 := g := bbgcdgen([b1,b2],[x1,x2,x3,x4,x5,x6,x7
          ,x8,y1,y2,y3,y4,y5,y6,y7,y8],
          rand(1..100),rand(1..10000));
proc()
local pol,ra,rb,rc,dl,gb,dgb,ffegen;
    pol := [b1,b2];
    ra := *** TABLE CORRESPONDING TO a2...an ***
    rb := *** TABLE CORRESPONDING TO b1...bn ***
    rc := *** TABLE CORRESPONDING TO c3...cr ***
    dl := *** TABLE CORRESPONDING TO d1...dr ***
    gb := 89856+25344*X;
    dgb := 1;
    ffegen := proc()
              local t;
                  _seed := irem(427419669081*
                          _seed,999999999989);
                  t := _seed;
                  irem(t,10000)+1
              end;
    RETURN(bbgcdaux(pol,ra,rb,rc,[args],dl,gb,
          dgb,ffegen))
end

time   18.68   words   3416888
O4 := g(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
                                          0

time   31.68   words   5950846
```

Figure 3: The GCD black box of b1 and b2

Note that the call to `bbgcdaux` in the returned procedure in Figure 3 executes the statements of Step A and Step B in the algorithm description and that the generated program for the GCD is independent of input black boxes except for the used constants.

This paper is to be considered as the pilot to the project of building a subsystem for manipulating multivariate polynomials and rational functions in black box representation. The DAGWOOD system (Freeman et al. 1988) realizes the straight-line program representation in Lisp with a natural interface to Macsyma. By virtue of our Maple prototype we have demonstrated that the black box representation can be implemented entirely within a general purpose computer algebra system. There are, however, two additional goals that we wish to accomplish:

(i) The black box programs produced by our algorithms, such as the program for the GCD, should be compilable.

(ii) Several algorithms manipulating black boxes, such as the sparse interpolation procedures, should be distributable over a network of compute nodes for parallel execution.

We set goal (i) as the result of efficiency considerations. A. Lobo has implemented a generic sparse linear system solver in C++ using black box representations for matrices over finite fields (cf. Díaz et al. 1993, §3). the linear system solver accesses the black box matrix as two function arguments, one to initialize static data in the fashion of a constructor in object oriented programming, and one to perform the necessary matrix times vector products, in the fashion of a member functions. The black boxes produced by our algorithms have a similar format. Thus a universal compilable evaluation procedure for the GCD, similar to `bbgcdaux` above, can be written provided we allow function arguments. Such is our approach for building our system based on the C++ or the $A^\sharp$ language (Watt et al. 1994). The problem remains that our GCD procedure is then not callable from Maple, say, due to the difficulty of passing a Maple function argument to a foreign library function. It is our hope that the OpenMath interface language will address this problem.

Distribution of the black box programs can be performed by our DSC system (Díaz et al. 1991). We have the facility to dynamically compile a program that has been constructed

```
O1 := b1 := bbdmgen(vandermonde([x1,x2,x3,x4,x5,x6,x7,x8]),
O1 := [x1,x2,x3,x4,x5,x6,x7,x8,y1,y2,y3,y4,y5,y6,y7,y8]);
proc()
local i,AA,p;
    AA := *** TABLE CORRESPONDING TO THE VANDERMONDE MATRIX ***
    p := [x1,x2,x3,x4,x5,x6,x7,x8,y1,y2,y3,y4,y5,y6,y7,y8];
    AA := subs(seq(p[i] = args[i],i = 1 .. nargs),op(AA));
    RETURN(det(op(AA)))
end

time   0.13   words   7360
O2 := b2 := bbdmgen(vandermonde([x1,x2,y3,y4,y5,y6,y7,y8]),
O2 := [x1,x2,x3,x4,x5,x6,x7,x8,y1,y2,y3,y4,y5,y6,y7,y8]);
proc()
local i,AA,p;
    AA := *** TABLE CORRESPONDING TO THE VANDERMONDE MATRIX ***
    p := [x1,x2,x3,x4,x5,x6,x7,x8,y1,y2,y3,y4,y5,y6,y7,y8];
    AA := subs(seq(p[i] = args[i],i = 1 .. nargs),op(AA));
    RETURN(det(op(AA)))
end

time   0.11   words   5342
```

Figure 4: Black boxes for the determinants of Vandermonde matrices

by one of our algorithms either locally or separately on each compute node. We also have built an interface to a general purpose system, namely to Maple (Chan et al. 1994). We plan to provide full parallel functionality for our future system, in particular for the new sparse interpolation algorithms.

5 Literature Cited

Ben-Or, M. and Tiwari, P., "A deterministic algorithm for sparse multivariate polynomial interpolation," *Proc. 20th Annual ACM Symp. Theory Comp.*, pp. 301–309 (1988).

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

Brown, W. S. and Traub, J. F., "On Euclid's algorithm and the theory of subresultants," *J. ACM* **18**, pp. 505–514 (1971).

Chan, K. C., Diaz, D., and Kaltofen, E., "A distributed approach to problem solving in Maple," in *Maple V: Mathematics and its Application*, Proceedings of the Maple Summer Workshop and Symposium, edited by R. J. Lopez; Birkhäuser, Boston, pp. 13–21, 1994.

Char, B. W., Geddes, K. O., and Gonnet, G. H., "GCDHEU: Heuristic polynomial GCD algorithm based on integer GCD computation," *J. Symbolic Comput.* **7**/1, pp. 31–48 (1989).

Díaz, A., Hitz, M., Kaltofen, E., Lobo, A., and Valente, T., "Process scheduling in DSC and the large sparse linear systems challenge," in *Proc. DISCO '93*, Springer Lect. Notes Comput. Sci. **722**, edited by A. Miola; pp. 66–80, 1993. To appear in *J. Symbolic Comput.*

Díaz, A., Kaltofen, E., Schmitz, K., and Valente, T., "DSC A System for Distributed Symbolic Computation," in *Proc. 1991 Internat. Symp. Symbolic Algebraic Comput.*, edited by S. M. Watt; ACM Press, pp. 323-332, 1991.

Drexler, F. J., "Eine Methode zur Berechnung sämtlicher Lösungen von Polynomgleichungssystemen," *Numer. Math.* **29**, pp. 45–58 (1977). (In German.)

Freeman, T. S., Imirzian, G., Kaltofen, E., and Lakshman Yagati, "DAGWOOD: A system for manipulating polynomials given by straight-line programs," *ACM Trans. Math. Software* **14**/3, pp. 218–240 (1988).

von zur Gathen, J., Karpinski, M., and Shparlinski, I., "Counting curves and their projections," *Manuscript*, June 1994.

Grigoriev, D. and Karpinski, M., "A zero-test and an interpolation algorithm for the shifted sparse polynomials," in *Proc. AAECC-10*, Springer Lect. Notes Comput. Sci. **673**, edited by G. Cohen, T. Mora, and O. Moreno; pp. 162–169, 1993.

Grigoriev, D. Yu., Karpinski, M., and Singer, M. F., "Fast parallel algorithms for sparse multivariate polynomial interpolation over finite fields," *SIAM J. Comput.* **19**/6, pp. 1059–1063 (1990).

Grigoriev, D. Yu. and Lakshman Y. N., "Algorithms for computing sparse shifts for multivariate polynomials," *Manuscript*, Drexel University, Philadelphia, Pennsylvania, December 1994.

Kaltofen, E., "Sparse Hensel lifting," *Proc. EUROCAL '85, Vol. 2, Springer Lec. Notes Comp. Sci.* **204**, pp. 4–17 (1985).

Kaltofen, E., "Greatest common divisors of polynomials given by straight-line programs," *J. ACM* **35**/1, pp. 231–264 (1988).

Kaltofen, E., "Factorization of polynomials given by straight-line programs," in *Randomness and Computation*, Advances in Computing Research **5**, edited by S. Micali; JAI Press, Greenwhich, Connecticut, pp. 375–412, 1989.

Kaltofen, E. and Lakshman Yagati, "Improved sparse multivariate polynomial interpolation algorithms," *Proc. ISSAC '88, Springer Lect. Notes Comput. Sci.* **358**, pp. 467–474 (1988).

Kaltofen, E., Lakshman Y. N., and Wiley, J. M., "Modular rational sparse multivariate polynomial interpolation," in *Proc. 1990 Internat. Symp. Symbolic Algebraic Comput.*, edited by S. Watanabe and M. Nagata; ACM Press, pp. 135–139, 1990.

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

Lakshman Y. N. and Saunders, B. D., "On computing sparse shifts for univariate polynomials," in *Proc. Internat. Symp. Symbolic Algebraic Comput. ISSAC '94*, edited by J. von zur Gathen and M. Giesbrecht; ACM Press, New York, N. Y., pp. 108–113, 1994.

Lakshman Y. N. and Saunders, B. D., "Sparse polynomial interpolation in non-standard bases," *SIAM J. Comput.*, to appear (1995).

Mansour, Y., "Randomized interpolation and approximation of sparse polynomials," in *Proc. ICALP 92*, Springer Lect. Notes Comput. Sci. **623**, edited by W. Kuich; 1992.

Moses, J. and Yun, D. Y. Y., "The EZ-GCD algorithm," *Proc. 1973 ACM National Conf.*, pp. 159–166 (1973).

Rayes, M. O., Wang, P. S., and Weber, K., "Parallelization of the sparse modular GCD algorithm for multivariate polynomials on shared memory multiprocessors," in *Proc. Internat. Symp. Symbolic Algebraic Comput. ISSAC '94*, edited by J. von zur Gathen and M. Giesbrecht; ACM Press, New York, N. Y., pp. 66–73, 1994.

Schwartz, J. T., "Fast probabilistic algorithms for verification of polynomial identities," *J. ACM* **27**, pp. 701–717 (1980).

Wang, P. S., "The EEZ-GCD algorithm," *SIGSAM Bulletin* **14/2**, pp. 50–60 (1980).

Watt, S. M., Broadbery, P. A., Dooley, S. S., Iglio, P., Morrison, S. C., Steinbach, J. M., and Sutor, R. S., "A first report on the $A^\sharp$ compiler," in *Proc. Internat. Symp. Symbolic Algebraic Comput. ISSAC '94*, edited by J. von zur Gathen and M. Giesbrecht; ACM Press, New York, N. Y., pp. 25–31, 1994.

Zippel, R., "Probabilistic algorithms for sparse polynomials," *Proc. EUROSAM '79, Springer Lec. Notes Comp. Sci.* **72**, pp. 216–226 (1979).

Zippel, R., "Interpolating polynomials from their values," *J. Symbolic Comput.* **9**/3, pp. 375–403 (1990).