A Sparse Modular GCD Algorithm for Polynomials over Algebraic Function Fields *

Seyed Mohammad Mahdi Javadi
School of Computing Science
Simon Fraser University
Burnaby, B.C. Canada.
sjavadi@cecm.sfu.ca.

Michael Monagan
Department of Mathematics
Simon Fraser University
Burnaby, B.C. Canada.
mmonagan@cecm.sfu.ca.

ABSTRACT

We present a first sparse modular algorithm for computing a greatest common divisor of two polynomials $f_1, f_2 \in L[x]$ where L is an algebraic function field in $k \geq 0$ parameters with $r \geq 0$ field extensions. Our algorithm extends the dense algorithm of Monagan and van Hoeij from 2004 to support multiple field extensions and to be efficient when the gcd is sparse. Our algorithm is an output sensitive Las Vegas algorithm.

We have implemented our algorithm in Maple. We provide timings demonstrating the efficiency of our algorithm compared to that of Monagan and van Hoeij and with a primitive fraction-free Euclidean algorithm for both dense and sparse gcd problems.

Categories and Subject Descriptors: I.1.2 [Symbolic and Algebraic Manipulation]: Algorithms – Algebraic algorithms;

General Terms: Algorithms, Theory.

Keywords: gcd algorithms, sparse interpolation, algebraic function fields.

1. INTRODUCTION

Let $F = \mathbb{Q}(t_1, \ldots, t_k)$, $k \geq 0$. For i, $1 \leq i \leq r$, let $m_i(z_1, \ldots, z_i) \in F[z_1, \ldots, z_i]$ be monic and irreducible over $F[z_1, \ldots, z_{i-1}]/\langle m_1, \ldots, m_{i-1} \rangle$. Let $L = F[z_1, \ldots, z_r]/\langle m_1, \ldots, m_r \rangle$. L is an algebraic function field in k parameters $t_1, \ldots, t_k$ (this also includes number fields). Let f_1 and f_2 be non-zero polynomials in $L[x]$ and let g be their monic gcd. Our problem is, given f_1 and f_2 to compute g or an associate (scalar multiple) of g. In a computer system, computations with polynomials over algebraic function fields arise, for example, when one solves non-linear polynomial equations involving parameters.

*Supported by NSERC of Canada and the MITACS NCE of Canada

ISSAC'07, July 29–August 1, 2007, Waterloo, Ontario, Canada.

One way to compute g would be to use the Euclidean algorithm. If one does this in $L[x]$ there is an expression swell in F and one must compute with large fractions in F. In [7] Moreno Maza and Rioboo show how to avoid arithmetic with fractions in F for univariate gcd computation modulo a triangular set of polynomials which includes L as a special case. However, the $k+1$ dimensional expression swell that occurs in the coefficients in F makes their algorithm very slow even for inputs of moderate degree.

In [4], Monagan and van Hoeij presented ModGcd, a first modular GCD algorithm for computing gcds over algebraic function fields presented with one field extension. Their algorithm uses a dense interpolation method and hence the time complexity is at least $O(d^k)$ where d bounds the degree of g in the k parameters.

Our algorithm, presented in Section 3, which we call *SparseModGcd* is a sparse modular GCD algorithm. Similar to ModGcd, our algorithm uses rational number reconstruction (see [9, 8]) and a variable at a time rational function reconstruction (see [4, 8]) to recover the coefficients of the gcd one parameter at a time. Like ModGcd, our algorithm is probabilistic and output sensitive. It uses trial division to prove the correctness of the result and the number of images it computes depends on the size of the gcd g and not f_1 and f_2 which may be much larger then g.

Unlike ModGcd, our algorithm uses a sparse interpolation to reduce the number of images needed to interpolate g when g is sparse. Zippel's sparse interpolation in [10], was originally developed for gcd computation in $\mathbb{Z}[x_1, \ldots, x_n]$. It is not efficient when the gcd is not monic in the main variable x_1. In [6], Wittkopf, *et al.* presented two techniques to make Zippel's sparse interpolation efficient for non-monic gcds. We modify one of these techniques for $L[x]$.

Our paper is organized as follows. In Section 2 we present an example of our algorithm showing the main flow of our algorithm. It also shows the reader how Zippel's sparse interpolation works, and illustrates some of the problems that may occur, the use of rational function reconstruction and other key design features of the algorithm. We then identify all problems that can occur and provide the theory for the basis of our algorithm which is presented in Section 3.

In Section 4 we compare Maple implementations of our algorithm with ModGcd and with a primitive fraction-free algorithm (see Appendix) on both dense and sparse problem sets to demonstrate its efficiency. Some of our problem sets are multivariate polynomials, that is, problems in $L[x_1, \ldots, x_n]$. In order to use our algorithm for multivari-

ate polynomial inputs we use the same method described by Monagan and van Hoeij in [4], namely, compute the content $c \in L[x_2, \ldots, x_n]$ of the gcd g from the inputs f_1 and f_2, then apply our algorithm to f_1/c and f_2/c treating the polynomial variables $x_2, \ldots, x_n$ as parameters.

2. AN EXAMPLE

Similar to ModGcd (see [4]), our algorithm works with *primitive associates* of the inputs and the minimal polynomials and computes the primitive associate of the gcd g, which we now define.

DEFINITION 1. *Let $D = \mathbb{Z}[t_1, ..., t_k]$. A non-zero polynomial in $D[z_1, \ldots, z_r, x]$ is said to be primitive wrt $(z_1, \ldots, z_r, x)$ if the gcd of its coefficients in D is 1. Let f be non-zero in $L[x]$ where L is the algebraic function field previously defined. The denominator of f is the polynomial $den(f) \in D$ of least total degree in $(t_1, \ldots, t_k)$ and with smallest integer content such that $den(f)f$ is in $D[z_1, \ldots, z_r, x]$. The primitive associate $\check{f}$ of f is the associate of $den(f)f$ which is primitive in $D[z_1, \ldots, z_r, x]$ and has positive leading coefficient in a term ordering.*

EXAMPLE 1. *Let $f = 3tx^2 + 6tx/(t^2-1) + 30tz/(1-t)$ where $m_1(z) = z^2 - t$. Here $f \in L[x]$ where $L = \mathbb{Q}(t)[z]/\langle z^2 - t\rangle$ is an algebraic function field in one parameter t. We have $\text{den}(f) = t^2 - 1$ and $\check{f} = \text{den}(f)f/(3t) = (t^2 - 1)x + 2x - 10z(t+1)$.*

We demonstrate our algorithm on the following example where we use s and t for parameters instead of t_1 and t_2.

EXAMPLE 2. *Let $m_1(t, s, z) = z^2 + st - 1$, $f_1(t, s, z, x) = g \times a$, and $f_2(t, s, z, x) = g \times b$ where $g = x^2 + \frac{1}{2t}zx + st + 5$, $a = 3zx - 3x + 5t + 10s - 1$ and $b = 3zx - 3x + 10s + 4$. Thus g is the monic gcd of f_1 and f_2.*

Our algorithm will compute and output $\check{g} = 2tx^2 + zx + 2st^2 + 10t$. It first computes $\check{f}_1 = 2tf_1$, $\check{f}_2 = 2tf_2$ and $\check{m}_1 = z^2 + st - 1$ to clear fractions from the inputs. To compute $\check{g}$, it computes $\check{g}/2$ modulo a sequence of primes. Let $p_1 = 17$ be the first prime. To compute $\check{g}/2 \bmod 17$ our algorithm will compute

$$g_1(t, s, z, x) = \check{g}/(2t) \bmod 17 = x^2 + \frac{9}{t}zx + \frac{t}{1}s + \frac{5}{1}$$

by interpolating t from points in $\mathbb{Z}_{17}$. Looking at g_1 we can see that two points are sufficient to interpolate the coefficients $\{9/t, t/1, 5/1\}$ in $\mathbb{Z}_{17}(t)$ if we know the degrees of their numerators and denominators. But we don't, and we don't have good bounds for them in general. We will use a probabilistic algorithm called maximal quotient rational function reconstruction – see [8, 5]. This algorithm will determine these coefficients using three points (one more than the minimum necessary) with high probability. Amongst the rational functions $n(t)/d(t)$ with $d(t)$ monic which interpolate the points, it computes one of minimal degree.

Let $t = 5$ be our first evaluation point. To compute $g_1(5, s, z, x) = x^2 + 12zx + 5s + 5 \bmod 17$, we apply our algorithm recursively to $\check{f}_1(5, s, z, x)$, $\check{f}_2(5, s, z, x)$ and $\check{m}_1(5, s, z)$. To interpolate s we need three points. Let $s = 4$ be the first point. To compute $g_1(5, 4, z, x) = x^2 + 12zx + 8 \bmod 17$ our algorithm computes the monic gcd of $\check{f}_1(5, 4, z, x)$ and $\check{f}_2(5, 4, z, x)$ modulo the ideal $\langle \check{m}_1(5, 4, z), p_1\rangle = \langle z^2 + 2, 17\rangle$ using the monic Euclidean algorithm (see [4]). That is, we execute the monic Euclidean algorithm over the finite ring $\mathbb{Z}_{17}[z]/(z^2 + 2)$. This finite ring is not field since $z^2 + 2 = (z+7)(z+10)$ in $\mathbb{Z}_{17}[z]$. But the monic Euclidean algorithm does not encounter a zero divisor on this input and hence (see Theorem 1) its output is still good.

Suppose we have computed $g_1(5, s, z, x) = x^2 + 12zx + 5s + 5 \bmod 17$ recursively. To interpolate t in $g_1(t, s, z, x)$ we need more points. Suppose we pick $t = 10$. Now comes Zippel's sparse interpolation assumption; we assume that $g_1(5, s, z, x)$ is of the correct form, that is,

$$g_1(10, s, z, x) = x^2 + Azx + Bs + C = g_f(s, z, x)$$

for some constants A, B and C. To determine A, B, C, we use sparse interpolation. We choose $s = 13$ and compute

$$g_1(10, 13, z, x) = x^2 + 6zx + 16$$

using the monic Euclidean algorithm with $t = 10, s = 13$. Equating coefficients in $x^i z^j$ of this with those in the assumed form $g_f(13, z, x)$ we obtain

$$\{A = 6, 13B + C = 16\} \bmod 17.$$

Notice that because of the form of g_f, we have two independent linear systems, one of dimension 1 and the other of dimension 2 so we need another equation to solve for B and C. Choosing $s = 14$ we obtain

$$g_1(10, 14, z, x) = x^2 + 6zx + 9,$$

hence $14B + C = 9 \bmod 17$. Solving for A, B, C we obtain $A = 6, B = 10$ and $C = 5$ and hence

$$g_1(10, s, z, x) = x^2 + 6zx + 10s + 5.$$

which completes the sparse interpolation of $g_1(10, s, z, x)$. Interpolating the images of g_1 at $t = 5$ and $t = 10$ yields

$$G = x^2 + (9t + 1)xz + st + 5.$$

Applying maximal quotient rational function reconstruction to the coefficients of G in $x^i z^j s^k$ modulo $(t-5)(t-10)$ fails so we need another evaluation point for t.

If we choose the next evaluation point $t = 1$, and use $s = 6$, applying the Euclidean algorithm to $\check{f}_1(1, 6, z, x)$ and $\check{f}_2(1, 6, z, x)$ modulo $\langle \check{m}_1(1, 6, z), 17\rangle$ we obtain

$$x^3 + (13z + 4)x^2 + (2z + 1)x + 10z + 10.$$

The degree of this image in x is greater than the degree of the assumed form. In fact no matter, what value of s we choose, this always happens. We say $t = 1$ is an *unlucky* evaluation point. It must be detected and rejected. Suppose we try $t = 11$ instead. Using sparse interpolation we obtain

$$g_1(11, s, z, x) = x^2 + 7zx + 11s + 5.$$

Interpolating the images of g_1 at $t = 5, 10$ and 11 we get

$$G = x^2 + (10t^2 + 12t + 8)xz + st + 5.$$

This time maximal quotient rational function reconstruction when applied to the coefficients of G in $x^i z^j s^k$ modulo $(t-5)(t-10)(t-11)$ succeeds and outputs

$$h = x^2 + \frac{9}{t}zx + \frac{t}{1}s + \frac{5}{1}.$$

Hence $\check{h} = tx^2 + 9zx + st^2 + 5t$. Since $\check{h}|\check{f}_1 \bmod p_1$ and $\check{h}|\check{f}_2 \bmod p_1$, we have $\check{h} = \gcd(\check{f}_1, \check{f}_2) \bmod p_1$ and we are done with our first prime p_1.

Now we apply the rational number reconstruction (see [8]) to $\{9, 1, 5\}$, the integer coefficients of g_1 modulo $p_1 = 17$. It fails because our prime is not big enough, so we need another image. We choose $p_2 = 11$. Let $g_2 = \gcd(\check{f}_1, \check{f}_2) \mod p_2$. Assuming that g_1 is of the correct form, we know that

$$g_2 = tx^2 + Hzx + (Ist^2 + Jt).$$

where we have fixed the leading coefficient of g_2 to be 1. Again we use sparse interpolation. We need two images to determine I and J. We choose two random evaluations for $(s, t) \in \mathbb{Z}_{11}^2$. We obtain $H = 6, I = 1$ and $J = 5$, hence

$$g_2 = tx^2 + 6zx + st^2 + 5t.$$

Now we apply Chinese Remaindering to g_1 and g_2 modulo primes $p_1 = 17$ and $p_2 = 11$. This results in

$$G = tx^2 + 94zx + st^2 + 5t \bmod 13 \times 11.$$

Again, we apply rational number reconstruction to $\{94, 1, 5\}$, the integer coefficients of G modulo $143 = 13 \times 11$. This time is succeeds and we obtain

$$h = tx^2 + \frac{1}{2}zx + st^2 + 5t,$$

hence

$$\check{h} = 2h = 2tx^2 + zx + 2st^2 + 10t.$$

Since $\check{h}|\check{f}_1$ and $\check{h}|\check{f}_2$, $\check{h} = \gcd(\check{f}_1, \check{f}_2)$ and we are done.

Remark: If the leading coefficient of $\check{g}$ in x ($2t$ in our example) was a sum of two or more terms, the sparse interpolation method used in the example would not work. This is called *Normalization Problem* (or leading coefficient problem). We will use one of the solutions introduced in [6].

2.1 Problems

In the example we encountered an unlucky evaluation point. There are several other problems that may arise depending on the primes and the evaluation points that the algorithm chooses, including the possibility of hitting a zero divisor while using the Euclidean algorithm to compute univariate images of the gcd. Here we identify all problems. We follow the terminology of van Hoeij and Monagan [4].

Bad primes and bad evaluation points.

DEFINITION 2. *A prime p is said to be a* bad prime *if the leading coefficient of $\check{f}_1$ or $\check{f}_2$ wrt x or any $\check{m}_i$ wrt z_i vanishes mod p. Similarly an evaluation point $t_j = \alpha$ is called a* bad evaluation point *if the degree of $\check{f}_1$ or $\check{f}_2$ wrt x or any $\check{m}_i$ wrt z_i decreases after evaluating at this point.*

EXAMPLE 3. *Suppose $\check{f}_1 = 28tx^3 + 19ztx + 2t^2 + 10, \check{f}_2 = 52zx^2 + 10x + zt^3 - t$ and $m(z) = (s-1)z^2 + 3$. Here $p_1 = 2, p_2 = 7$ and $p_3 = 13$ are bad primes. Also $t = 0$ and $s = 1$ are bad evaluation points.*

The good thing about bad primes and bad evaluation points is that they can be ruled out in advance.

Unlucky primes and unlucky evaluation points.

DEFINITION 3. *A prime p is said to be* unlucky *if $\check{g}_p = \gcd(\check{f}_1, \check{f}_2) \mod p$ has higher degree in x than the gcd $\check{g}$. Similarly an evaluation point $t_j = \alpha$ is said to be* unlucky *if $\gcd(\check{f}_1(\alpha), \check{f}_2(\alpha)) \bmod p$ has higher degree in x than $\check{g}$.*

EXAMPLE 4. *Consider the input polynomials*

$\check{f}_1 = (x+z)(x+17t+t^2+z)$ and $\check{f}_2 = (x+t)(x+t^2+z)$.

Here $\check{g} = 1$ but $\check{g}_{17} = \gcd(\check{f}_1, \check{f}_2) \mod 17 = x+t^2+z$ which obviously has higher degree than $\check{g}$, so $p = 17$ is an unlucky prime. Similarly $t = 0$ is an unlucky evaluation point.

Unlucky primes must be avoided if $\check{g}$ is to be correctly reconstructed. One can show provided p is large, the probability p being unlucky is low. Unlike bad primes, unlucky primes can not be detected and discarded in advance. Brown in [1] showed how to do this in a way that is efficient for $\mathbb{Z}[x]$; whenever images of the gcd do not have the same degree, one keeps only those images of smallest degree and discard the others. The same solution works here. See Theorem 1.

Zero Divisors and the Euclidean Algorithm

Recall that a non-zero element α of a ring R is a zero divisor if there exists a non-zero element $\beta \in R$ s.t. $\alpha\beta = 0$. When we are using the Euclidean algorithm to compute the gcd of $f_1(\alpha_1, \ldots, \alpha_k, x)$, $f_2(\alpha_1, \ldots, \alpha_k, x)$ ($\alpha_1, \ldots, \alpha_k$ are the evaluation points) modulo a prime p, we might encounter a zero divisor, in which case the Euclidean algorithm fails (see [3, 4]). The bigger the prime p is, the smaller the chance of hitting a zero divisor would be.

EXAMPLE 5. *Let $f_1 = (z+2t)x^2+tx+z$, $f_2 = zx^3+tx^2+(z-2)x+8t$ and $m(z) = z^2 - t$. Suppose we choose the first prime $p = 7$. If we evaluate the inputs at $t = 2$ we obtain $f_1' = (z-3)x^2 + 2x + z$ and $f_2' = zx^3 + 2x^2 + (z-2)x + 2$. When we run the Euclidean algorithm on the inputs f_1' and f_2' we hit a zero divisor while trying to invert $\mathrm{lc}(f_1') = z - 3$. Note $z^2 - 2 = (z-3)(z+3) \bmod 7$.*

The solution to the problem with zero divisors is described in the next section.

THEOREM 1. *Let $f_1, f_2 \in L[x]$ be two non-zero polynomials where $L = F[z_1, \ldots, z_r]/\langle m_1, \ldots, m_r \rangle$ and $F = \mathbb{Q}(t_1, \ldots, t_k)$. Let $\check{g} = \gcd(\check{f}_1, \check{f}_2)$. Let p be a prime and $\alpha = (t_1 = \alpha_1, \ldots, t_k = \alpha_k)$. Suppose that the monic Euclidean algorithm applied to $\check{f}_1(\alpha, x)$ and $\check{f}_2(\alpha, x)$ modulo p does not encounter a zero divisor and outputs g_p (monic in x). If α is not a bad evaluation point and p is not a bad prime, $\deg_x(g_p) \geq \deg_x(\check{g})$. Moreover if $\deg_x(g_p) = \deg_x(\check{g})$ then $g_p = \mathrm{monic}(\check{g}(\alpha, x) \bmod p)$ and we say α is a good evaluation point and p is a good prime.*

Proof: See Monagan and van Hoeij [4]. The difference is the number of field extensions but this has no significant change in the proof in [4].

Theorem 1 and Brown's method for detecting unlucky primes and unlucky evaluation points tells us how to get good primes and good evaluation points which will give us univariate images from which we will reconstruct the multivariate coefficients of $\check{g}$ in $\mathbb{Z}[t_1, ..., t_k]$. There are three additional problems (and we claim only three) to address. The first is due our using sparse interpolation.

Missing Terms

DEFINITION 4. *A prime p is said to introduce* missing terms *if any term of $\check{g}$ vanishes modulo p. Similarly an evaluation point $t_k = \alpha$ is said to introduce* missing terms *if any coefficient of $\check{g}$ in $\mathbb{Z}_p[t_k]$ vanishes at $t_k = \alpha$.*

EXAMPLE 6. *Let* $\check{g} = (t^2 - s + 1)x^3 + 70zx^2 + (13t + 26)$. *Here the primes* $2, 5, 7$ *and* 13 *introduce missing terms (the first three cause the second term to vanish and the last one makes the last term to vanish). The prime* $p = 11$ *does not introduce missing terms, but when we are computing the image of the gcd modulo this prime,* $t = 9$ *makes the last term vanish.*

Unfortunately primes (or evaluation points) which introduce missing terms can not be avoided in advance. We detect an incorrect assumed form g_f as follows. If during the sparse interpolation we compute an image with the same degree as the assumed form g_f in x but with terms in $(x, z_1, \ldots, z_r)$ not present in g_f, the assumed form g_f is wrong. The following example shows what will happen if the assumed form is wrong but the terms are the same.

EXAMPLE 7. *Suppose* $\check{g} = x + tz + 13z$. *Suppose the first prime used is* $p_1 = 13$. *Then the assumed form for* $g(x, z, t)$ *will be* $g_f = x + Btz$. *Now suppose we pick the second prime* $p_2 = 11$ *and attempt sparse interpolation, equating* $\check{g}(t) = g_f(t)$ *for different* t. *For* $t = 1$ *we obtain* $x+3z = x+Bz \implies B = 3$. *For* $t = 2$ *we obtain* $x + 4z = x + 2Bz \implies B = 2$, *an inconsistency.*

We argue that an incorrect assumed form g_f will be identified with high probability if we use one more evaluation point than the minimum needed to solve for the unknowns in the assumed form. Once an incorrect assumed form is identified, because we do not know whether it was the current evaluation point or a previous evaluation point or the current prime that caused the missing terms, we must allow our algorithm to restart with a new prime.

Remark 2: Suppose $\check{g} = (7s^2 + 9t + 1)x^2 + z$. Here $p = 7$ is not bad but we still can not use it because the image of the gcd computed modulo $p = 7$ does not have the correct leading term (see [4]). Note that in our algorithm $p = 7$ causes a missing term and this is treated in the same way we treat missing terms.

Normalization problem

An image of the gcd computed modulo a prime p is unique up to a scaling factor in the integers mod p. This causes a complication in the sparse interpolation when $\mathrm{lc}_x\check{g}$ has more than one term in the parameters.

EXAMPLE 8. *Let* $z = \sqrt{s + t}$. *Suppose the input polynomials are* $\check{f}_1 = (x - s + 1)\check{g}$ and $\check{f}_2 = (x + t + s)\check{g}$ *where*

$$\check{g} = (15s + t)x^2 + 12s^2xz + 40st$$

is the gcd of $\check{f}_1$ *and* $\check{f}_2$. *Suppose we have computed*

$$\check{g}_7 = (s + t)x^2 + 5s^2xz + 5st,$$

our first image modulo $p_1 = 7$. *So our assumed form is*

$$g_f = (As + Bt)x^2 + Cs^2xz + Dst$$

for some constants A, B, C *and* D. *Now we want to compute the next image of the gcd modulo* $p_2 = 11$. *Consider the evaluation point* $\alpha = (s = 2, t = 1)$, *we have*

$$\check{g}_{21} = \gcd(\check{f}_1(2, 1), \check{f}_2(2, 1)) = 1 \cdot x^2 + 9xz + 4.$$

The problem is that this image is unique *up to a* scaling factor μ. *That is*

$$\mu(x^2 + 9xz + 4) = (2A + B)x^2 + 4Cxz + 2D, \quad (*)$$

but we do not know what μ *is. If we knew the leading coefficient of* $\check{g}$, $\mathrm{lc}_x(\check{g}) = 15s + t$, *then we could easily compute*

$$\mu = \mathrm{lc}_x(\check{g}(2, 1)) \bmod 11 = 9.$$

Unfortunately there is no easy way of computing $\mathrm{lc}_x\check{g}$.

Suppose g_f is the assumed form of the gcd and we have computed $h = \gcd(\check{f}_1(\alpha), \check{f}_2(\alpha)) \bmod p$. If g is monic then we have the equation $h = g_f(\alpha)$, otherwise we let μ be an unknown in (*). Because μ is a new unknown, we may need more images to construct a consistent system of linear equations. The above solution was first introduced by Wittkopf, *et al.* in [6]. To see how this affects the structure of the linear systems consider the problem of finding a gcd which

$$\begin{bmatrix} c & c & & & & & 1 & \\ c & c & & & & & & 1 \\ & & c & c & & & c & \\ & & c & c & & & & c \\ & & & & c & c & c & \\ & & & & c & c & & c \end{bmatrix} \begin{bmatrix} a_1 \\ a_0 \\ b_1 \\ b_0 \\ c_1 \\ c_0 \\ 1 \\ \mu_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Figure 1: Structure of the linear system for multiple scaling case.

looks like

$$g_f = (a_1s^2 + a_0)x^2 + (b_1st^2 + b_0)x + (c_1t^2 + c_0)z.$$

Here we need two images. The linear system of equations has a structure shown in Figure 1 where all entries not shown are zero and $\mu_1 = 1$. The solution can be easily computed by solving a number of smaller subsystems corresponding to the rectangular blocks of non-zero entries augmented with the multiplier columns. With this method, the solution expense turns out to be of the same order as the case where $\check{g}$ is monic.

Unlucky Contents

DEFINITION 5. *For a polynomial* $f = a_nx^n + \cdots + a_1x + a_0$ *with* $a_i \in R$ *for* $0 \le i \le n$, *the* content

$$\mathrm{cont}_x(f) = \gcd(a_0, a_1, \ldots, a_n) \in R.$$

A prime p, *is said to introduce an* unlucky content *if for two input polynomials* f_1, f_2 *with gcd* $g = \gcd(f_1, f_2)$, $\mathrm{cont}_x(\check{g}) = 1$ *but* $\mathrm{cont}_x(\check{g} \bmod p) \neq 1$. *Similarly an evaluation point* $t = \alpha$ *is said to introduce an unlucky content if* $\mathrm{cont}_x(\check{g}) = 1$ *but* $\mathrm{cont}_x(\check{g}(\alpha)) \neq 1$.

EXAMPLE 9. *Suppose* $\check{g} = (12s + t)x + (s + 12t)z$. *We have* $\mathrm{cont}_x(\check{g}) = \gcd(12s+t, s+12t) = 1$. *But for* $p = 11$ *we obtain* $cont_x(\check{g} \bmod p) = s + t$. *Hence* $p = 11$ *introduces an unlucky content and, for any prime* p, *the evaluation points* $t = 0$ *and* $s = 0$ *introduce unlucky contents.*

Suppose during sparse interpolation we choose our assumed form, g_f, based on an image which is computed modulo a prime (or evaluation point) which introduced an unlucky content, e.g. $p_1 = 11$ in our example. Then the assumed form g_f will have different terms in $x, z_1, ..., z_r$, $t_1, ..., t_k$ than in $\check{g}$. This will, with high probability, result in an inconsistent linear system during sparse interpolation.

Suppose instead that our assumed form g_f is correct but it is a subsequent prime or evaluation point that introduces an unlucky content. This can lead to an under determined linear system.

EXAMPLE 10. *Consider* $f_1 = f_2 = zx + t + 1$ *where* $m(z) = z^2 - t - 14$. *Here* $g = x + (t+1)/(t+14)z$ *and hence* $\check{g} = (t+14)x + (t+1)z$ *thus* 13 *introduces an unlucky content* t. *Suppose our first prime is* $p_1 \neq 13$ *and we obtain the correct assumed form* $g_f = (At+B)x + (Ct+D)z$. *Suppose our second prime is* $p_2 = 13$ *and we perform a sparse interpolation in* t *using* $t = 1, 2, 3, ...$ *Since* g_f *is not monic in* x *we will equate* $g_f(t) = \mu_t \check{g}(t)$ *and solve for* $A, B, C, D, \mu_1, \mu_2, ...$ *with* $\mu_1 = 1$. *For* $t = 1, 2, 3$ *we obtain the following equations modulo* 13.

$$(A + B)x + (C + D)z \equiv \mu_1(x + z),$$
$$(2A + B)x + (2C + D)z \equiv \mu_2(x + z),$$
$$(3A + B)x + (3C + D)z \equiv \mu_3(x + z).$$

Equating coefficients of $x^i z^j$ *we obtain the following linear system:* $A+B = \mu_1$, $2A+B = \mu_2$, $3A+B = \mu_3$, $C+D = 1$, $2C + D = 1$, $3C + D = 1$. *The reader may verify that this system, with* $\mu_1 = 1$, *is not determined, and also, adding further equations, for example, from* $t = 4$, *does not make the system determined.*

If our primes are sufficiently large and our evaluation points are chosen at random, then primes and evaluation points which introduce unlucky contents are rare. Because also, in $L[x]$, we cannot easily identify them in advance (it is not necessarily true that $\mathrm{lc}_x \check{g}$ divides $\mathrm{lc}_x \check{f}_1$) we will detect them through their effect. We will assume that if the linear system in sparse interpolation is not determined, an unlucky content is present, and we will design our algorithm to fail back to the point where the unlucky content was introduced.

It is also possible that the linear system is under determined because of the evaluation points chosen in sparse interpolation. For example, for $\check{g} = x + (t^3 - t)z$ with assumed form $g_f = x + (At^3 + B)z$, evaluation points $t = 0, 1, -1$ do not constrain the system. Thus when we mistakenly assume that this is because of an unlucky content, we may waste some useful work.

3. ALGORITHM SPARSEMODGCD

We now present the SparseModGcd algorithm. This modular GCD algorithm first calls subroutine M which computes the gcd in $L[x]$ from a number of images in $L_p[x]$. Subroutine P which is called by subroutine M computes the gcd in $L_p[x]$ using both dense and sparse interpolations. Finally subroutine S, which stands for Sparse Interpolation and is called by subroutine P, does the sparse interpolation.

Algorithm SparseModGcd
Input: $f_1, f_2 \in L[x]$ and $m_1, \ldots, m_r \in F[z_1, \ldots, z_r]$ where $F = \mathbb{Q}(t_1, \ldots, t_k)$ s.t. $\mathrm{cont}_x(g) = 1$.
Output: $\check{g}$, where g is the monic gcd of f_1 and f_2 in $L[x]$.

1. Call Subroutine M with input $\check{f}_1$, $\check{f}_2$ and $\check{m}_1, \ldots, \check{m}_r$.

Subroutine M
Input: $f_1, f_2 \in D[z_1, ..., z_r]/\langle m_1, ..., m_r \rangle [x]$ and $m_1, ..., m_r \in D[z_1, \ldots, z_r]$ were $D = \mathbb{Z}[t_1, \ldots, t_k]$.
Output: $\check{g}$, where g is the monic gcd of f_1 and f_2.

1. Set $n = 1, G = 0$, and the assumed form $g_f = 0$.
2. Take a new prime p that is not bad.
3. Let g_n be the output of subroutine **P** applied to p, (f_1, f_2) mod p, g_f, and $(m_1, ..., m_r)$ mod p.
4. If $g_n =$ "ZeroDivisor" or $g_n =$ "Unlucky" or $g_n =$ "UnluckyContent" then go back to step 2.
5. If $g_n =$ "BadForm" then go back to step 1.
6. If $g_n = 1$ then return 1.
7. If $G = 0$ then set $G = g_n$, $M = p$ and go to step 9.
8. Set $M = M \times p$ and combine g_n with $\{g_1, \ldots, g_{n-1}\}$ using Chinese remaindering to obtain G mod M.
9. Set $g_f = G$, $n = n + 1$.
10. Apply integer rational reconstruction to obtain h satisfying $h \equiv G \bmod M$. If this fails then go back to step 2.
11. Clear fractions in $\mathbb{Q}$: Set $h = \check{h}$.
12. Trial division: if $h | f_1$ and $h | f_2$ then return h, otherwise, go back to step 2.

Subroutine P
Input: p, f_1, f_2, $g_f \in D_p[z_1, \ldots, z_r]/\langle m_1, \ldots, m_r \rangle [x]$ and $m_1, \ldots, m_r \in D_p[z_1, \ldots, z_r]$.
Output: Either $\check{g}$, the primitive associate of the monic gcd of f_1 and f_2, or "ZeroDivisor" or "Unlucky" or "BadForm" or "UnluckyContent."

1. If k (the number of parameters) $= 0$ then output the result of the monic Euclidean algorithm applied to f_1, f_2 modulo $\langle m_1, ..., m_r, p \rangle$. If a zero divisor is encountered then output "ZeroDivisor".
2. If the assumed form $g_f = 0$ then go to step 4.
3. **Sparse Interpolation:** (we already know the form g_f of the gcd from subroutine M.)
   Return $g_n \in D_p[z_1, \ldots, z_r, x]$, the output of subroutine **S** applied to $p, f_1, f_2, m_1, \ldots, m_r$ and g_f.
4. Choose α_1 at random from $\mathbb{Z}_p$ that is not bad.
5. Let $g_1 \in \mathbb{Z}_p[t_1, \ldots, t_{k-1}][z_1, \ldots, z_r, x]$ be the output of subroutine **P** applied to $p, f_1, f_2, m_1, \ldots, m_r$ at $t_k = \alpha_1$ and assumed form $g_f = 0$.
6. If $g_1 = 1$ then return 1.
7. If $g_1 \in$ { "BadForm", "Unlucky", "UnluckyContent", "ZeroDivisor" } then return g_1.
8. Set $g_f = g_1$, $G = g_1$, $M = (t_k - \alpha_1)$, $n = 2$, $c = 1, d = 1, u = 1$.
9. **Main Loop:** Take a new evaluation point α_n *at random* from $\mathbb{Z}_p$ that is not bad.
10. Let $g_n \in \mathbb{Z}_p[t_1, \ldots, t_{k-1}][z_1, \ldots, z_r, x]$ be the output of subroutine **S** applied to $p, f_1, f_2, m_1, \ldots, m_r$ evaluated at $t_k = \alpha_n$ and g_f.
11. If $g_n =$ "BadForm" then return "BadForm".
12. If $g_n =$ "UnluckyContent" then set $c = c + 1$ and if $c > n$ return "UnluckyContent" else go to main loop.
13. If $g_n =$ "Unlucky" then set $u = u + 1$ and if $u > n$ then return "Unlucky", else go back to main loop.
14. If $g_n =$ "ZeroDivisor" then set $d = d + 1$ and if $d > n$ then return "ZeroDivisor", else go back to main loop.
15. If $g_n = 1$ then return 1.
16. Set $M = M \times (t_k - \alpha_n)$ and Chinese remainder g_n with $\{g_1, \ldots, g_{n-1}\}$ to obtain G mod $M(t_k)$.
17. Set $n = n + 1$.

18. Apply rational function reconstruction to coefficients of G to obtain $h \in \mathbb{Z}_p(t_k)\ [t_1, \ldots, t_{k-1}][z_1, \ldots, z_r, x]$ s.t. $h \equiv G \mod M(t_k)$. If this fails, go back to main loop.
19. Clear fractions in $\mathbb{Z}_p(t_k)$: Set $h = \check{h}$.
20. Trial division: if $h|f_1$ and $h|f_2$ then return h, otherwise, go back to main loop.

Subroutine S
Input: $p, f_1, f_2, g_f \in D_p[z_1, \ldots, z_r]/\ \langle m_1, \ldots, m_r \rangle\ [x]$ and $m_1, \ldots, m_r \in D_p[z_1, \ldots, z_r]$ where $D_p = \mathbb{Z}_p[t_1, \ldots, t_k]$.
Output: Either $\check{g}$, the primitive associate of the monic gcd of f_1 and f_2, or "BadForm" or "ZeroDivisor" or "Unlucky" or "UnluckyContent."

1. If k (the number of parameters) $= 0$ then call the monic Euclidean algorithm on f_1,f_2 and output the result.
2. Suppose the assumed form $g_f = \sum_i C_i T_i$ where T_i is a monomial in $(x, z_1, \ldots, z_r)$ and $C_i = \sum_j c_{ij} S_j$ where S_j is a monomial in parameters $t_1, \ldots, t_k$ with unknown c_{ij}.
3. Set U to be the minimum number of images needed – see below. (The algorithm uses one more image than U to detect a wrong assumed form g_f.)
4. Set $d = 1$, $u = 1, n = 1$.
5. While $n \leq U + 1$ do
   5.1. Take a new *random* evaluation point $\alpha_n = (t_1 = a_1, \ldots, t_k = a_k)$ in $\mathbb{Z}_p^k$ which is not bad.
   5.2. Let g_n be the output of the monic Euclidean algorithm applied to $f_1(\alpha_n)$, $f_2(\alpha_n)$ modulo $\langle m_1(\alpha_n)$,..., $m_r(\alpha_n), p\rangle$.
   5.3. If $g_n =$ "ZeroDivisor" then set $d = d + 1$ and if $d > n$ then return "ZeroDivisor" else go back to step 5.1.
   5.4. If $\deg_x(g_n) > \deg_x(g_f)$ then set $u = u + 1$ and if $u > n$ then return "Unlucky" else go back to step 5.1.
   5.5. If $\deg_x(g_n) < \deg_x(g_f)$ return "BadForm".
   5.6. If g_n has terms in $(x, z_1, \ldots, z_r)$ not present in the assumed form g_f then return "BadForm".
   5.7. Set $n = n + 1$.
6. Construct the system of $U + 1$ linear equations by equating $g_f(\alpha_n) = \mu_n g_n$ with μ_n unknown. Solve the linear system with $\mu_1 = 1$ to determine the c_{ij}s.
7. If the system is inconsistent, return "BadForm".
8. If the system is under determined, return "UnluckyContent".
9. Set $g_p = \sum_i (\sum_j c_{ij} S_j) T_i$.
10. Make $\mathrm{lc}_{x,t_1,\ldots,t_k}(g_p) = 1$ and return g_p.

Let n_i be the number of terms in the ith coefficient C_i of the assumed form g_f. The minimum number of images needed in the sparse interpolation (with multiple scaling factors) to determine the c_{ij}s is $\max(M, \left\lceil \frac{N-1}{T-1} \right\rceil)$ where T is the number of monomials in g_f in $(x, z_1, \ldots, z_r)$, N is the total number of monomials in g_f and $M = \max_i n_i$.

If the assumed form g_f in subroutine S is wrong, the linear system will most probably be inconsistent. It can, however, be consistent. If consistent, control passes back to subroutine P (or subroutine M) which will attempt rational function reconstruction (respectively, rational number reconstruction in subroutine M). If this succeeds the trial divisions prevent the algorithm from returning a wrong answer. Then subroutine P will call subroutine S at a new evaluation point with the same wrong assumed form. We argue that, provided p is sufficiently large, subroutine S will eventually get an inconsistent system and determine that the assumed form is wrong.

To treat zero divisors we use the same strategy used by Monagan and van Hoeij in ModGcd algorithm (see [4]). The variable d, in subroutine P, counts the number of times the Euclidean algorithm encounters a zero divisor. The case $d > n$ happens when the algorithm encounters a lot of zero divisors. This could relate to our choice of prime in subroutine M or a previous evaluation point in subroutine P. In this case subroutine P will quit. Note that if most evaluation points are good, and if subroutine P has already computed many good images, then the test $d > n$ prevents, with high probability, that an unlucky choice in Step 9 could cause a lot of useful work to be lost.

A similar strategy is also used in subroutine S for both zero divisors and unlucky primes to prevent useful work from being lost. The same strategy is also used in subroutine P to prevent useful work from being lost should the current evaluation point $t = \alpha_n$ in Step 10 introduce an unlucky content.

4. IMPLEMENTATION

Here we explain *trial division* which is a bottleneck in the implementation of SparseModGcd algorithm.

Trial Division

In Step 12 of subroutine M and Step 20 of subroutine P, the algorithm uses *trial division* to test whether it has computed the correct gcd. The only difference is that in subroutine P, the trial divisions take place in characteristic p. In [3] Monagan, van Hoeij presented an algorithm for doing trial divisions (in characteristic p) of polynomials in $\mathbb{Z}[z][x]$ modulo $m(z) \in \mathbb{Z}[z]$ which uses pseudo-division to avoid fractions and some gcds in $\mathbb{Z}$ to minimize growth of the integer coefficients. We can use the same idea for our algorithm, except that the coefficient ring is $D_p = \mathbb{Z}_p[t_1, \ldots, t_k]$ instead of $\mathbb{Z}$. The same algorithm can also be used for subroutine M with D_p replaced by D. Here we show how to extend it to treat multiple field extensions.

Algorithm Trial Division with Multiple Field Extensions
Input: $A, B \in D_p[z_1, ..., z_r]/\ \langle m_1, ..., m_r \rangle\ [x]$ and $m_1, ..., m_r \in D_p[z_1, ..., z_r]$, $B \neq 0$.
Output: True if $B|A$, False otherwise.

1. Set $m = \deg_x(A), n = \deg_x(B)$.
2. Set $d_1 = \deg_{z_1}(m_1), \ldots, d_r = \deg_{z_r}(m_r)$.
3. Set $l_b = \mathrm{lc}_x(B)$.
4. Set $l_{m_1} = \mathrm{lc}_{z_1}(m_1), \ldots, l_{m_r} = \mathrm{lc}_{z_r}(m_r)$.
5. Set $R = A$.
6. While $R \neq 0$ and $m \geq n$ do
   6.1. Set $l_R = \mathrm{lc}_x(R)$.
   6.2. Set $g = \gcd(\mathrm{cont}_{z_1,\ldots,z_r}(l_R), l_b)$ *mod p*.
   6.3. Set $l_R = l_R/g, s = l_b/g$.

6.4. Set $t = l_R x^{m-n}$.
6.5. Set $R = sR - tb$.
6.6. For i from 1 to r do
i. While $R \neq 0$ and $\deg_{z_i}(R) \geq d_i$ do
A. Set $l_R = \mathrm{lc}_{z_i}(R)$.
B. Set $g = \gcd(\mathrm{cont}_x(l_R), l_{m_i})$ *mod p*.
C. Set $l_R = l_R/g$.
D. Set $t = l_R z_i^{\deg_{z_i}(R)-d_i}$.
E. Set $R = (l_{m_i}/g)R - t m_i$.
6.7. Set $m = \deg_x(R)$.
7. If $R \neq 0$ then return False, otherwise, return True.

Note that $\deg_{z_j}(m_i) = 0$ if $j > i$. The outer loop reduces the degree of the remainder R in x. In the inner loops, for each i, the algorithm reduces the degree of R in z_i to be less than the degree of m_i in z_i.

4.1 Timings

We have compared Maple implementations of our SparseModGcd algorithm, the ModGcd algorithm of van Hoeij and Monagan [4] and a primitive fraction-free Euclidean algorithm (see Appendix) on three problem sets. The input polynomials have a sparse gcd in the first and third set and a dense gcd in the second problem set. All timings are in CPU seconds and were obtained using Maple 10 on a 64 bit AMD Opteron CPU @ 2.4 GHz, running Linux, using 31.5 bit primes.

SPARSE-1

Let $m(z) = z^3-(s+r)z^2-(t+v)^2z-5-3u$. For $n = 1, 2, \ldots, 10$, let $f_1 = a \times g$ and $f_2 = b \times g$ where

$$g = sx_1^n+tx_2^n+ux_3^n+\sum_{j=1}^{4}\sum_{i=0}^{n-1} r_{ij}^{(1)} z^{j-1}x_j^i+\sum_{w=[r,s,t,u,v]}\sum_{k=0}^{n} r_{w_k}^{(1)} w^k,$$

$$a = tx_1^n+ux_2^n+sx_3^n+\sum_{j=1}^{4}\sum_{i=0}^{n-1} r_{ij}^{(2)} z^{j-1}x_j^i+\sum_{w=[r,s,t,u,v]}\sum_{k=0}^{n} r_{w_k}^{(2)} w^k,$$

$$b = ux_1^n+sx_2^n+tx_3^n+\sum_{j=1}^{4}\sum_{i=0}^{n-1} r_{ij}^{(3)} z^{j-1}x_j^i+\sum_{w=[r,s,t,u,v]}\sum_{k=0}^{n} r_{w_k}^{(3)} w^k$$

and each $r_{j_k}^{(i)}$ is a positive random integer less than 100. Thus we have 10 gcd problems, all with one field extension $m(z)$, five parameters r, s, t, u and v and four variables x_1, x_2, x_3 and x_4. Each input polynomial is of degree $2n$ in the first three variables and $2n-2$ in x_4. We wanted a data set of polynomials which are sparse but not too sparse.

Table 1 gives the running times for the three algorithms. In the first column, the numbers shown in parentheses are the percentages of the time which is spent computing univariate images of the gcd using the Euclidean algorithm. Since the gcd g in this case is sparse (g has $9n+3$ terms and $\deg(g) = n$ in any of x_1, x_2 and x_3), a better performance is expected from SparseModGcd. The data demonstrates this clearly.

DENSE-1

Let $m(z) = z^2 - sz - 3$. Suppose g, a and b are three randomly chosen polynomials in x_1, x_2, s and z of total degree n which are dense. That is, the term $x_1^{d_1}x_2^{d_2}s^{d_3}z^{d_4}$ with $0 \leq d_1 + d_2 + d_3 + d_4 \leq n$ is present in each of these three polynomials. So each has exactly $\sum_{i=0}^{n}\binom{i+4}{4}$ terms before we reduce by $m(z)$. For $n = 1, 2, \ldots, 10, 15$, let $f_1 = g \times a$ and $f_2 = g \times b$. Since the gcd g is dense, ModGcd algorithm is expected to perform better.

n	SparseModGcd	ModGcd	PPRS
1	0.38 (51.72%)	8.70	3.83
2	0.98 (62.15%)	114.78	> 2000
3	2.03 (67.06%)	879.26	NA
4	3.82 (73.40%)	> 2000	NA
5	6.58 (75.17%)	NA	NA
6	11.03 (77.76%)	NA	NA
7	17.31 (79.78%)	NA	NA
8	26.55 (81.31%)	NA	NA
9	39.02 (82.31%)	NA	NA
10	54.54 (82.34%)	NA	NA

Table 1: Timings (in CPU seconds) for SPARSE-1 (NA means not attempted)

n	SMGCD	(EA,TDIV)	ModGcd	PPRS
1	0.031	(67.74%,6.45%)	0.029	0.002
2	0.070	(58.57%,21.43%)	0.058	0.384
3	0.150	(60.66%,24.67%)	0.141	367.234
4	0.308	(44.80%,34.42%)	0.307	> 2000
5	0.497	(40.44%,42.86%)	0.557	NA
6	0.901	(44.73%,43.40%)	1.272	NA
7	1.516	(38.19%,49.93%)	2.091	NA
8	2.443	(34.67%,53.70%)	3.244	NA
9	3.847	(30.93%,58.33%)	5.024	NA
10	6.120	(27.78%,63.71%)	7.437	NA
15	59.878	(29.48%,66.40%)	50.228	NA

Table 2: Timings (in CPU seconds) for DENSE-1 (NA means not attempted)

Table 2 shows the running times of the three algorithms for this set of problems. In the second column, the first number shown in parentheses is the percentage of the time spent computing univariate images of the gcd and the second is the percentage spent doing trial division. The reader may see that SparseModGcd is very competitive with ModGcd even though the gcds are completely dense.

SPARSE-2

Let $T = [s, t, u, s, t, u, \ldots]$ and $Z = [z_1, z_2, z_1, z_2, \ldots]$.
Let $m_1(z_1) = z_1^2 - (t^2 + s)z_1 - 5 - 3u$, and
$m_2(z_2) = z_2^2 + (s - z_1)z_2 + 3t - u$.
For $n = 1, 2, \ldots, 10$ let $f_1 = a \times g$ and $f_2 = b \times g$ where

$$g = sx^n + \sum_{i=0}^{n-1} iT_{i+1}Z_{i+1}x^i,$$

$$a = tx^n + \sum_{i=0}^{n-1} iT_{i+2}Z_{i+2}x^i, \quad b = ux^n + \sum_{i=0}^{n-1} iT_{i+3}Z_{i+3}x^i.$$

So we have 10 gcd problems with two field extensions m_1 and m_2, three parameters s, t and u and one variable x. Each input polynomial is of degree $2n$ in x.

The timings for SparseModGcd and primitive PRS algorithms for this problem set are shown in Table 3. In the

n	SparseModGcd	PPRS
1	0.096 (87.50%)	0.004
2	0.229 (84.28%)	194.540
3	0.361 (87.53%)	> 2000
4	0.510 (88.82%)	NA
5	0.695 (90.37%)	NA
6	0.921 (91.42%)	NA
7	1.180 (90.34%)	NA
8	1.493 (90.69%)	NA
9	1.811 (93.04%)	NA
10	2.172 (93.42%)	NA

Table 3: Timings (in CPU seconds) for SPARSE-2 (NA means not attempted)

first column, the numbers shown in parentheses are the percentages of the time spent computing univariate images of the gcd. Since ModGcd does not support multiple field extensions, there are no timings for ModGcd. The data here demonstrates the superiority of SparseModGcd algorithm.

5. CONCLUDING REMARKS

We mention the GCD algorithm of Dahan *et al.* in [2] which computes the gcd of two univariate polynomials modulo a triangular set T of dimension zero over $\mathbb{Q}$, that is, not involving any parameters. Their algorithm is designed to treat also the case where the triangular set does not generate a prime ideal and consequently a zero divisor could be encountered in the Euclidean algorithm. It may still be desirable to compute a gcd if it exists. Here we briefly explain how our algorithm could be modified to treat zero divisors.

Suppose $L = F[z]/\langle m \rangle$ and m is reducible over F. Thus L is not a field. It is a commutative ring with zero divisors. If the Euclidean algorithm when applied to f_1 and f_2 would encounter a zero divisor (a factor of $m(z)$) then our algorithm, when it executes the Euclidean algorithm in Subroutine P and S will most likely encounter an image of that zero divisor. Thus our algorithm will most likely go into an infinite loop.

In principle, one may modify our algorithm to interpolate the zero divisor using sparse interpolation. Since one will not know whether the zero divisor exists over $\mathbb{Q}$ or is caused by an unlucky choice of prime or evaluation point, one simultaneously interpolates the gcd and zero divisor. Subroutines M and P would terminate when either the interpolated zero divisor divides $m(z)$ or the interpolated gcd divides f_1 and f_2.

The algorithm of Dahan *et al.* is also a sparse GCD algorithm. It uses Hensel lifting instead of sparse interpolation. To use Hensel lifting for GCD computation in $L[x]$, one would need to compute, and possibly also factor, $\alpha(t) \in \mathbb{Z}[t_1, ..., t_k]$, a multiple of the leading coefficient of $\check{g}$. For an algebraic function field L computing $\alpha(t)$ requires us to invert the leading coefficients of $\check{f}_1$ and $\check{f}_2$ (or compute resultants) which may introduce a severe expression swell. Such an algorithm would not be output sensitive. Another advantage of sparse interpolation over Hensel lifting is that it is more easy to parallelize.

6. REFERENCES

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

[2] X. Dahan, M. Moreno Maza, E. Schost, W. Wu and Y. Xie, Lifting Techniques for Triangular Decompositions. *Proceedings of ISSAC '05*, ACM Press, pp. 108–115, 2005.

[3] Mark van Hoeij and Michael Monagan, A modular GCD algorithm over number fields presented with multiple extensions. *Proceedings of ISSAC '02*, ACM Press, pp. 109–116, 2002.

[4] Mark van Hoeij and Michael Monagan. Algorithms for Polynomial GCD Computation over Algebraic Function Fields, *Proceedings of ISSAC '04*, ACM Press, pp. 297–304, 2004.

[5] Sara Khodadad and Michael Monagan. Fast Rational Function Reconstruction. *Proceedings of ISSAC '06*, ACM Press, pp. 184–190, 2006.

[6] J. de Kleine, M. Monagan and A. Wittkopf, Algorithms for the non-monic case of the sparse modular GCD algorithm. *Proceedings of ISSAC '05*, ACM Press, pp. 124–131, 2005.

[7] Marc Moreno Maza and Renaud Rioboo. Polynomial Gcd Computations over Towers of Algebraic Extensions. *Proceedings of AAECC-11*, Springer-Verlag, pp. 365–382, 1995.

[8] Michael Monagan, Maximal Quotient Rational Reconstruction: An Almost Optimal Algorithm for Rational Reconstruction. *Proceedings of ISSAC '04*, ACM Press, 243–249, 2004.

[9] P. S. Wang, M. J. T. Guy, J. H. Davenport. Early detection of true factors in Univariate Polynomial Factorization. *Proceedings of EUROCAL '83*, Springer-Verlag LNCS **162**, pp. 225–235.

[10] Richard Zippel, Probabilistic algorithms for sparse polynomials. *Proceedings of EUROSAM '79*, Springer-Verlag LNCS **72**, pp. 216–226, 1979.

Acknowledgment

We gratefully acknowledge the help of an anonymous referee who identified and fixed an error in our algorithm.

Appendix

The description and the Maple code for the primitive fraction-free algorithm that we used to compute a gcd in $L[x]$ for comparison with SparseModGcd algorithm is available from

`http://www.cecm.sfu.ca/CAG/code/sjavadiPPRS.pdf`

The algorithm is a modification of the fraction free algorithm of Maza and Rioboo [7], where, we make make pseudo-remainders and quasi-inverses primitive.