MATHEMATICS OF COMPUTATION
Volume 84, Number 296, November 2015, Pages 3013–3027
http://dx.doi.org/10.1090/mcom/2951
Article electronically published on April 15, 2015

THE USE OF BAD PRIMES IN RATIONAL RECONSTRUCTION

JANKO BÖHM, WOLFRAM DECKER, CLAUS FIEKER, AND GERHARD PFISTER

Abstract. A standard method for finding a rational number from its values modulo a collection of primes is to determine its value modulo the product of the primes via Chinese remaindering, and then use Farey sequences for rational reconstruction. Successively enlarging the set of primes if needed, this method is guaranteed to work if we restrict ourselves to "good" primes. Depending on the particular application, however, there may be no efficient way of identifying good primes.

In the algebraic and geometric applications we have in mind, the final result consists of an a priori unknown ideal (or module) which is found via a construction yielding the (reduced) Gröbner basis of the ideal. In this context, we discuss a general setup for modular and, thus, potentially parallel algorithms which can handle "bad" primes. A new key ingredient is an error tolerant algorithm for rational reconstruction via Gaussian reduction.

1. Introduction

Rational reconstruction, in conjunction with Chinese remaindering, provides a standard way of obtaining results over the rational numbers from results in characteristic $p > 0$. This is of particular use in the design of parallel algorithms and in situations where the growth of intermediate results matters. Classical applications are the computation of polynomial greatest common divisors (see [6, 11]) and Gröbner bases (see [2]). The goal of the modular Gröbner basis algorithm presented in [2] is to compute the Gröbner basis of an ideal *already known*. That is, the ideal is given by a finite set of generators. In contrast, there are constructions which yield *a priori unknown* ideals by finding their (reduced) Gröbner bases. Prominent examples are the computation of normalization and the computation of adjoint curves (see [3, 4]). Here, for the purpose of modularization, we expect that the respective construction applies to a given set of input data in characteristic zero as well as to the modular values of the input data. In such a situation, problems may arise through the presence of "bad" primes of different types.

Usually, a first step to resolve these problems is to show that the bad primes are "rare". Then the different types of bad primes are addressed. For example, a prime for which it is a priori clear that the modular construction does not work will simply be rejected. Depending on the application, however, there may be bad primes which can only be detected a posteriori, that is, after the true characteristic zero result has been found. For such an application, we must ensure that the reconstruction algorithm used will return the correct rational number even in the presence of bad primes. In this note, we derive such an algorithm. Based on this algorithm, we

Received by the editor July 16, 2012 and, in revised form, September 27, 2013 and March 24, 2014.

2010 *Mathematics Subject Classification.* Primary 13P10, 68W10; Secondary 52C05.

Key words and phrases. Rational reconstruction, Farey map.

describe a general scheme for computing ideals (or modules) in the algebraic and geometric applications we have in mind, addressing, in particular, how various types of bad primes are handled. This scheme has already been successfully used in the aforementioned papers [3, 4].

To begin, in Section 2, we recall the classical approach to rational reconstruction which is based on the lifting of modular to rational results by computing Farey preimages via Euclidean division with remainder. Section 3 contains a short discussion of the different types of bad primes. In Section 4, we present the new lifting algorithm which is based on Gaussian reduction, and discuss the resulting error tolerant reconstruction algorithm. Finally, in Section 5, we present our general scheme for applications in commutative algebra and algebraic geometry. We finish by giving an explicit example which involves a bad prime that can only be detected a posteriori.

To summarize, we focus on the presentation of a general setup for modular computations based on error tolerant rational reconstruction. We do not discuss implementation details or performance questions. In fact, for the applications we have in mind, the time used for rational reconstruction can be neglected in view of the more involved parts of the respective algorithms.

2. Reconstruction of a single rational number

We describe the reconstruction of a single unknown number $x \in \mathbb{Q}$. In practical applications, this number will be a coefficient of an object to be computed in characteristic 0 (for example, a vector, polynomial, or Gröbner basis). Here, we suppose that we are able to verify the correctness of the computed object (in some applications by a comparably easy calculation, in others by a bound on the size of the coefficients).

We use the following notation: Given an integer $N \geq 2$ and a number $x = \frac{a}{b} \in \mathbb{Q}$ with $\gcd(a, b) = 1$ and $\gcd(b, N) = 1$, the *value of x modulo N* is

$$x_N := \left(\frac{a}{b}\right)_N := (a + N\mathbb{Z})(b + N\mathbb{Z})^{-1} \in \mathbb{Z}/N\mathbb{Z}.$$

We also write $x \equiv r \bmod N$ if $r \in \mathbb{Z}$ represents x_N.

In what follows, we suppose that in the context of some application, we are given an algorithm which computes the value of the unknown number $x \in \mathbb{Q}$ modulo any prime p, possibly rejecting the prime. For reference purposes, we formulate this in the black box type Algorithm 1.

Algorithm 1. Black Box Algorithm $x \bmod p$

Input: A prime number p.

Output: `false` or an integer $0 \leq s \leq p - 1$ such that $x \equiv s \bmod p$.

Assumption: There are only finitely many primes p where the return value is `false`.

Once the values of x modulo each prime in a sufficiently large set of primes $\mathcal{P}$ have been computed, we may find x via a lifting procedure consisting of two steps: First, use Chinese remaindering to obtain the value of x modulo the product $N := \prod_{p \in \mathcal{P}} p$. Second, compute the preimage of this value under the *Farey rational map* which is defined as follows.

For an integer $B > 0$, set

$$F_B = \left\{ \frac{a}{b} \in \mathbb{Q} \mid \gcd(a,b) = 1,\ 0 \le a \le B,\ 0 < |b| \le B \right\},$$

and for $m \in \mathbb{Z}/N\mathbb{Z}$, let

$$\mathbb{Q}_{N,m} = \left\{ \frac{a}{b} \in \mathbb{Q} \mid \gcd(a,b) = 1,\ \gcd(b,N) = 1,\ \left(\frac{a}{b}\right)_N = m \right\}$$

be the set of rational numbers whose value modulo N is m. Then $\mathbb{Q}_N = \bigcup_{m=0}^{N-1} \mathbb{Q}_{N,m}$ is a subring of $\mathbb{Q}$ with identity. If B is an integer with $B \le \sqrt{(N-1)/2}$, then the *Farey map*,

$$\varphi_{B,N} : F_B \cap \mathbb{Q}_N \to \mathbb{Z}/N\mathbb{Z}, \quad \frac{a}{b} \mapsto \left(\frac{a}{b}\right)_N,$$

is well defined and injective (but typically not surjective). To obtain the injective map with the largest possible image for a given N, we tacitly suppose in what follows that B is chosen as large as possible for N.

Algorithm 2 will return $\varphi_{B,N}^{-1}(\overline{r})$ if $\overline{r}$ is in the image of the Farey map, and `false` otherwise (see, for example, [5, 10–12]).

Algorithm 2. Farey Preimage

Input: Integers $N \ge 2$ and $0 \le r \le N-1$.

Output: `false` or a rational number $\frac{a}{b}$ with $\gcd(a,b) = 1$, $\gcd(b,N) = 1$, $\frac{a}{b} \equiv r \bmod N$, $0 \le a \le \sqrt{(N-1)/2}$, $0 < |b| \le \sqrt{(N-1)/2}$.

1: $(a_0, b_0) := (N, 0)$, $(a_1, b_1) := (r, 1)$, $i := -1$
2: **while** $2a_{i+2}^2 > N - 1$ **do**
3: $\quad i := i + 1$
4: $\quad$ divide a_i by a_{i+1} to find q_i, a_{i+2}, b_{i+2} such that

$$(a_i, b_i) = q_i(a_{i+1}, b_{i+1}) + (a_{i+2}, b_{i+2})$$

$\quad$ and $0 \le a_{i+2} < a_{i+1}$
5: **if** $2b_{i+2}^2 \le N - 1$ **and** $\gcd(a_{i+2}, b_{i+2}) = 1$ **then**
6: $\quad$ **return** $\frac{a_{i+2}}{b_{i+2}}$
7: **return** `false`

Combining Algorithm 2 with Chinese remaindering as indicated above, we get the classical reconstruction Algorithm 3.

Note that Algorithm 3 works correctly since we suppose that our Black Box Algorithm 1 either returns `false` or a correct answer. For most applications, however, there exist primes p which are bad in the sense that the algorithm under consideration returns a wrong answer modulo p which can only be detected a posteriori. In this note, we show that if there are only finitely many such primes, they can just be ignored. More precisely, we show that in Algorithm 3, we may call the black box type Algorithm 4 instead of Algorithm 1, provided we then call the lifting Algorithm 5 from Section 4 instead of Algorithm 2.

Algorithm 3. Reconstruction of a Rational Number

Input: Algorithm 1 and a way to verify that a computed number equals x.
Output: x.

```
1: N := 1, r := 0
2: p := 2
3: loop
4:    let s be the return value of Algorithm 1 applied to p
5:    if s = false then
6:       continue with Step 13
7:    find 1 = eN + fp and set r := rfp + seN, N := Np
8:    let y be the return value of Algorithm 2 applied to N and r
9:    if y = false then
10:      continue with step 13
11:   if y = x then
12:      return y
13:   p :=NextPrime(p)
```

Algorithm 4. Black Box Algorithm x mod p With Errors

Input: A prime number p.
Output: `false` or an integer $0 \leq s \leq p-1$.

Assumption: There are only finitely many primes p where the return value is either `false` or satisfies $x \not\equiv s \bmod p$.

3. Types of bad primes for modular algorithms

In this section, we suppose that we are given an algorithm implementing a construction which applies in any characteristic, together with a set of input data over the rationals.

We call a prime p *good* for the given algorithm and input data if the algorithm applied to the modulo p values of the input returns the reduction of the characteristic zero result. Otherwise, the prime is called *bad*. In what follows, to make the discussion in the subsequent sections more clear, we specify various types of bad primes and describe their influence on the design of algorithms.

A prime p is bad of *type-1* if the modulo p reduction of the characteristic zero input is either not defined or for obvious reasons not a valid input for the algorithm. For example, if the input is a polynomial, type-1 primes arise from prime factors of a denominator of a coefficient. Type-1 primes are harmless with regard to rational reconstruction as they can be detected and, thus, rejected from the beginning, at no additional cost.

For a bad prime p of *type-2*, it turns out only in the course of the construction that the computation in characteristic p cannot be carried through. For example, an algorithm for inverting a matrix will not work for a prime dividing the determinant. Since, typically, the determinant is not known, the failure will only turn out eventually. Type-2 primes waste computation time, but with regard to rational reconstruction they are detected before the Chinese remainder step and do, thus, not influence the final lifting.

Consider an invariant whose characteristic zero value coincides with the characteristic p value for all good primes p, and suppose that this value is known a priori. Moreover, assume that the algorithm computes this invariant for a prime p at some point of the construction. Then p is bad of *type-3* if the value in characteristic p differs from the expected one. Like type-2 primes, bad primes of type-3 waste computation time for computing modular results which will then be discarded, but do not influence the final lifting. Examples of possible invariants are the dimension or the degree or of a variety. Note that the computation of an invariant for detecting a type-3 prime may be expensive. Dropping the computation of the invariant in the design of the algorithm, if possible, will turn a type-3 prime into a prime of different type. This includes primes of type-5 below.

Now suppose that some invariant associated to the modular output is computed by the algorithm, and that the a priori *unknown* characteristic zero value of this invariant coincides with the characteristic p value for all good primes p. Then a prime is bad of *type-4* if this invariant does not have the correct value. Such a prime cannot be detected a priori. However, if there are only finitely many such primes, they can be eliminated with arbitrarily high probability by a majority vote over several primes. Type-4 bad primes may occur, for example, in modular Gröbner basis computations, where we use the set of minimal generators of the leading ideal as an invariant for voting. Handling type-4 primes is expensive not only since we waste computation time, but also since we have to store the modular results for the majority vote. Again, in the setup of Section 5, such primes are eventually discarded, and then do not enter the Chinese remainder step. Using additional theoretic results on the invariant, it may be possible to avoid the majority vote. If, for example, it is known that the invariant is minimal for good primes, then we may always vote for the smaller result. The degree of a univariate polynomial gcd is an invariant of this type, likewise the δ-invariant of an algebraic curve.

Bad primes other than those discussed so far are called bad primes of *type-5*. This includes primes which cannot be discovered by any known means without knowledge of the characteristic zero result. Example 5.12 below shows that type-5 bad primes indeed occur in algebraic geometry. Type-5 bad primes enter the Chinese remainder step and are, thus, present during the final lifting process. Considering Algorithm 3, calling Black Box Algorithm 4 instead of Algorithm 1, we will be in a situation where always either Algorithm 2 or the comparison $x = y$ will return `false`. As a result, the algorithm will not terminate.

Due to their nature, bad primes hardly ever create a *practical* problem. Typically, there are only very few bad primes for a given instance, and these will not be encountered if the primes used are chosen at random. On the other hand, for some of the modern algorithms in commutative algebra, we have no *theoretical* argument eliminating type-5 bad primes. Hence, we need error tolerant reconstruction, which ensures termination provided there are only finitely many bad primes.

4. Reconstruction with bad primes

To design our error tolerant reconstruction algorithm, we turn rational reconstruction into a lattice problem.

To begin with, given an integer $N \geq 2$, we define the subset $C_N \subseteq \mathbb{Z}/N\mathbb{Z}$ of elements applied to which Algorithm 5 below will return a rational number (and not `false`). Let C_N be the set of all $\overline{r} \in \mathbb{Z}/N\mathbb{Z}$ such that there are integers $u, v \in \mathbb{Z}$

with $u \geq 0$, $v \neq 0$, and $\gcd(u, v) = 1$ which satisfy the following condition:

$$\text{there is an integer } q \geq 1 \text{ with } q|N \text{ and such that} \quad u^2 + v^2 < \tfrac{N}{q^2} \quad \text{and} \quad u \equiv vr \bmod \tfrac{N}{q}. \tag{1}$$

In Lemma 4.2 below, we will prove that the rational number $\frac{u}{v} = \frac{uq}{vq} \in \mathbb{Q}$ is uniquely determined by condition (1). Hence, we have a well-defined map

$$\psi_N : C_N \to \mathbb{Q}.$$

Note that the image of the Farey map $\varphi_{B,N}$, with $B = \left\lfloor \sqrt{(N-1)/2} \right\rfloor$, is contained in C_N: If $\overline{r} \in \operatorname{im}(\varphi_{B,N})$, then $\varphi_{B,N}^{-1}(\overline{r})$ satisfies condition (1) with $q = 1$. Furthermore, $\varphi_{B,N}^{-1}(\overline{r}) = \psi_N(\overline{r})$.

Typically, the inclusion $\operatorname{im}(\varphi_{B,N}) \subseteq C_N$ is strict:

Example 4.1. For $N = 2 \cdot 13$, we have $B = 3$, hence

$$\operatorname{im}(\varphi_{B,N}) = \left\{\overline{0}, \overline{1}, \overline{2}, \overline{3}, \overline{8}, \overline{9}, \overline{17}, \overline{18}, \overline{23}, \overline{24}, \overline{25}\right\},$$

and the rational numbers which can be reconstructed by Algorithm 2 are the elements of

$$F_B \cap \mathbb{Q}_N = \left\{0, \pm 1, \pm 2, \pm 3, \pm\frac{1}{3}, \pm\frac{2}{3}\right\}.$$

On the other hand,

$$C_N = \left\{\overline{r} \mid 0 \leq r \leq 25,\ r \neq 5, 21\right\},$$

and Algorithm 5 will reconstruct the rational numbers in

$$\psi_N(C_N) = \left\{0, \pm 1, \pm 2, \pm 3, \pm 4, \pm\frac{1}{2}, \pm\frac{1}{3}, \pm\frac{2}{3}, \pm\frac{4}{3}\right\}.$$

Note that the denominator of $\frac{1}{2} = \psi_N(7) = \psi_N(20)$ is not coprime to N. In both cases, $q = 2$: We have $1 \equiv 2 \cdot 7 \bmod 13$ and $1 \equiv 2 \cdot 20 \bmod 13$.

Now, fix $0 \leq r \leq N - 1$ such that $\overline{r} \in C_N$, and consider the lattice $\Lambda = \Lambda_{N,r} := \langle (N, 0), (r, 1) \rangle$ of discriminant N. Let u, v, q correspond to $\overline{r}$ as in condition (1). Then $(uq, vq) \in \Lambda_{N,r}$.

Lemma 4.2. *With notation as above, all $(x, y) \in \Lambda$ with $x^2 + y^2 < N$ are collinear. That is, they define the same rational number $\frac{x}{y}$.*

Proof. Let $\lambda = (x, y)$, $\mu = (c, d) \in \Lambda$ be vectors with $x^2 + y^2, c^2 + d^2 < N$. Then $y\mu - d\lambda = (yc - xd, 0) \in \Lambda$, so $N|(yc - xd)$. Since $|yc - xd| < N$, by Cauchy–Schwarz, we get $yc = xd$, as claimed. □

Next, consider integers $N', M \geq 2$, with $\gcd(M, N') = 1$, and such that $N = N'M$. Let $a \geq 0, b \neq 0$ be integers such that $\gcd(b, N') = 1$, and let $a \equiv bs \bmod N'$, with $0 \leq s \leq N' - 1$. Let $0 \leq t \leq M - 1$ be another integer, and let $0 \leq r \leq N - 1$ be the Chinese remainder lift satisfying $r \equiv s \bmod N'$ and $r \equiv t \bmod M$. In practical applications, we think of N' and M as the product of good and bad primes, respectively. By the following lemma, Algorithm 5 below applied to N and r will return a/b independently of the possibly "wrong result" t, provided that $M \ll N'$.

Lemma 4.3. *With notation as above, suppose that* $(a^2+b^2)M < N'$. *Then, for all* $(x,y) \in \Lambda = \langle (N,0),(r,1)\rangle$ *with* $(x^2+y^2) < N$, *we have* $\frac{x}{y} = \frac{a}{b}$. *Furthermore, if* $\gcd(a,b) = 1$ *and* (x,y) *is a shortest nonzero vector in* Λ, *we also have* $\gcd(x,y)|M$.

Proof. From $a \equiv bs \bmod N'$, we get $a - bs = k_1 N'$ for some k_1. Moreover, $s \equiv r \bmod N'$ gives $r = s + k_2 N'$. Now $(aM, bM) - bM(r,1) = (aM - brM, 0)$ and $aM - brM = M(a-br) = M(a-b(s+k_2N')) = M(a-bs) - k_2bN = k_1N - k_2bN$, thus $(aM, bM) \in \Lambda$. Since $(a^2+b^2)M < N'$, Lemma 4.2 gives $\frac{a}{b} = \frac{aM}{bM} = \frac{x}{y}$ for all $(x,y) \in \Lambda$ such that $(x^2+y^2) < N$.

For the second statement, write $A := (aM, bM)$ and $X := (x,y)$. By Lemma 4.2, there is a $\lambda = \frac{s}{t} \in \mathbb{Q}$, with $\gcd(s,t) = 1$, and such that $\lambda X = A$. The Euclidean Algorithm gives integers e, f with $et + sf = 1$, hence

$$\frac{X}{t} = (et+sf)\frac{X}{t} = eX + fA \in \Lambda.$$

Since X is a shortest vector in the lattice, it follows that $t = \pm 1$, hence $A = \pm sX$. Since $\gcd(a,b) = 1$, we conclude that $\gcd(x,y)|M$. □

The use of Lemma 4.3 is twofold. From a theoretical point of view, it allows us to ignore type-5 bad primes in the design of modular algorithms—as long as there are only finitely many of them. This makes the design of modular algorithms much simpler. From a practical point of view, it allows us to avoid expensive computations of invariants to eliminate bad primes of any type. Moreover, factorizing the gcd of the components of a shortest lattice element can help us to identify bad primes (see Example 4.5 below).

Lemma 4.3 yields the correctness of both the new lifting Algorithm 5 and the resulting reconstruction Algorithm 3, calling Black Box Algorithm 4 instead of Algorithm 1, and lifting Algorithm 5 instead of Algorithm 2. In applications, the termination can be based either on the knowledge of a priori bounds on the height of $\frac{x}{y}$ or on an a posteriori verification of the result. It should be mentioned that both methods are used: some problems allow for easy verification, while others yield good bounds.

Remark 4.4. Algorithm 5, which is just a special case of Gaussian reduction, will always find a shortest vector in the lattice generated by $(N,0)$ and $(r,1)$. Moreover, $b_i \neq 0$ for all $i > 0$ since in every step the vector (a_i, b_i) gets shorter and, hence, cannot be equal to $(N,0)$.

Even though Algorithm 5 looks more complicated than Algorithm 2, the bit–complexity of both algorithms is the same: $O(\log^2 N)$. See the discussion in [9, Section 3.3]. Experiments show that the runtime actually differs by a small constant factor, which is strongly dependent on the specific implementation.

In most applications, as described in Section 5, the runtime of the rational reconstruction is negligable compared to that of the total algorithm.

Example 4.5. We reconstruct the rational number $\frac{13}{12}$ using the modulus

$$N = 38885 = 5 \cdot 7 \cdot 11 \cdot 101.$$

With notation as above, $a = 13$, $b = 12$, $r = 22684$, and the Farey bound is

$$B = \left\lfloor \sqrt{(N-1)/2} \right\rfloor = 139.$$

Algorithm 5. Error Tolerant Lifting

Input: Integers $N \geq 2$ and $0 \leq r \leq N-1$.
Output: $\psi_N(\overline{r})$ if $\overline{r} \in C_N$ and `false` otherwise.

1: $(a_0, b_0) := (N, 0)$, $(a_1, b_1) := (r, 1)$, $i := -1$
2: **repeat**
3: $\quad i = i + 1$
4: $\quad$ set
$$q_i = \left\lfloor \frac{\langle (a_i, b_i), (a_{i+1}, b_{i+1}) \rangle}{\|(a_{i+1}, b_{i+1})\|^2} \right\rceil$$
5: $\quad$ set
$$(a_{i+2}, b_{i+2}) = (a_i, b_i) - q_i(a_{i+1}, b_{i+1})$$
6: **until** $a_{i+2}^2 + b_{i+2}^2 \geq a_{i+1}^2 + b_{i+1}^2$
7: **if** $a_{i+1}^2 + b_{i+1}^2 < N$ **then**
8: $\quad$ **return** $\frac{a_{i+1}}{b_{i+1}}$
9: **else**
10: $\quad$ **return** `false`

Algorithm 2 applied to this data will correctly return $\frac{13}{12}$. Similarly for Algorithm 5 which generates the sequence

$$\begin{aligned}
(38885, 0) &= 2 \cdot (22684, 1) + (-6483, -2), \\
(22684, 1) &= -3 \cdot (-6483, -2) + (3235, -5), \\
(-6483, -2) &= 2 \cdot (3235, -5) + (-13, -12), \\
(3235, -5) &= -134 \cdot (-13, -12) + (1493, -1613).
\end{aligned}$$

Now we make bad primes enter the picture. Consider the Chinese remainder isomorphism

$$\chi : \mathbb{Z}/5\mathbb{Z} \times \mathbb{Z}/7\mathbb{Z} \times \mathbb{Z}/11\mathbb{Z} \times \mathbb{Z}/101\mathbb{Z} \to \mathbb{Z}/38885\mathbb{Z}.$$

The preimage of $\overline{r} = \left(\frac{13}{12}\right)_N$ is

$$\chi^{-1}(\overline{r}) = (\overline{4}, \overline{4}, \overline{2}, \overline{60}).$$

That is, $\overline{r}$ is the solution to the simultaneous congruences

$$\begin{aligned}
x &\equiv 4 \bmod 5, \\
x &\equiv 4 \bmod 7, \\
x &\equiv 2 \bmod 11, \\
x &\equiv 60 \bmod 101.
\end{aligned}$$

If we make 101 a bad prime by changing the congruence $x \equiv 60 \bmod 101$ to $x \equiv 61 \bmod 101$, we obtain

$$\chi(\overline{4}, \overline{4}, \overline{2}, \overline{61}) = \overline{16524}.$$

Algorithm 5 then computes

$$\begin{aligned}(38885, 0) &= 2 \cdot (16524, 1) + (5837, -2),\\ (16524, 1) &= 3 \cdot (5837, -2) + (-987, 7),\\ (5837, -2) &= 6 \cdot (-987, 7) + (-85, 40),\\ (-987, 7) &= 10 \cdot (-85, 40) + (-137, 393).\end{aligned}$$

Hence the output $\frac{85}{-40} = \frac{17}{8} \neq \frac{13}{12}$ is not the desired lift. The reason for this is that 101 is not small enough compared to its cofactor in N. Algorithm 2, on the other hand, returns `false` since the reduction process will also terminate with $(85, -40)$ and these numbers are not coprime. Note that Algorithm 6 in Section 5 below will detect an incorrect lift either by the procedure pTest (with a very high probability) or the subsequent verification step over the rationals (carried through only if pTest returns `true`). As a consequence, in both cases, the set of primes will be enlarged (without discarding previous results). Eventually, the good primes will outweigh the bad ones and Algorithm 5, when called from Algorithm 6, will return the correct lift.

For example, replace the congruence $x \equiv 4 \bmod 7$ by $x \equiv 2 \bmod 7$, so that

$$\chi(\overline{4}, \overline{2}, \overline{2}, \overline{60}) = \overline{464}.$$

Then Algorithm 5 yields

$$\begin{aligned}(38885, 0) &= 84 \cdot (464, 1) + (-91, -84),\\ (464, 1) &= -3 \cdot (-91, -84) + (191, -251),\end{aligned}$$

and terminates with the correct lift

$$\frac{91}{84} = \frac{13}{12}.$$

Algorithm 2, on the other hand, will again return `false` since the reduction also terminates with the numbers $(91, 84)$ which are not coprime.

Since

$$(13^2 + 12^2) \cdot 7 < 5 \cdot 11 \cdot 101,$$

Lemma 4.3 shows that 7 is small enough compared to its cofactor in N. Hence, the wrong result 2 modulo the bad prime 7 does not influence the result of the lift. In fact, all other possible congruences modulo 7 will lead to the same output. Note that the bad prime can be detected as $\gcd(91, 84, N) = 7$. Furthermore, note that in the example the lifting process involving the bad prime requires fewer steps than the process relying on good primes only.

5. A setup for applications in algebra and geometry

In this section, we discuss a general computational setup for applications in commutative algebra and algebraic geometry which requires error tolerance. A setup of this type occurs, for example, when computing normalization or when computing adjoint curves. See [3, 4] and Example 5.11 below.

To begin, fix a global monomial ordering $>$ on the monoid of monomials in the variables $X = \{X_1, \ldots, X_n\}$. Consider the polynomial rings $W = \mathbb{Q}[X]$ and, given an integer $N \geq 2$, $W_N = (\mathbb{Z}/N\mathbb{Z})[X]$. If $T \subseteq W$ or $T \subseteq W_N$ is a set of polynomials, then denote by $\mathrm{LM}(T) := \{\mathrm{LM}(f) \mid f \in T\}$ its set of leading monomials. If $f \in W$ is a polynomial such that N is coprime to any denominator of a coefficient of f,

then its *reduction modulo* N is the polynomial $f_N \in W_N$ obtained by mapping each coefficient x of f to x_N as described in Section 2. If $H = \{h_1, \ldots, h_s\} \subset W$ is a Gröbner basis such that N is coprime to any denominator in any h_i, set $H_N = \{(h_1)_N, \ldots, (h_s)_N\}$. If $J \subseteq W$ is any ideal, its *reduction modulo* N is the ideal

$$J_N = \langle f_N \mid f \in J \cap \mathbb{Z}[X] \rangle \subseteq W_N.$$

NOTATION: *From now on, let $I \subset W$ be a fixed ideal.*

Remark 5.1. For practical purposes, I is given by a set of generators. Fix one such set $f_1, \ldots, f_r$. Then we realize the reduction of I modulo a prime p via the following equality which holds for all but finitely many primes p:

$$I_p = \langle (f_1)_p, \ldots, (f_r)_p \rangle \subseteq W_p.$$

More precisely, when running the modular Algorithm 6 described below, we incorporate the following: if one of the $(f_i)_p$ is not defined (that is, p is bad of type-1 for the given set of generators), we reject the prime[1]. If all $(f_i)_p$ are defined, we work with the ideal on the right-hand side instead of I_p. Note that it is possible to detect primes with $I_p \neq \langle (f_1)_p, \ldots, (f_r)_p \rangle$ (which are hence of type-3). Indeed, I_p can be found using Gröbner bases (see [1, Cor. 4.4.5] and [2, Lem. 6.1]). However, we suggest skipping this computation: finitely many bad primes will not influence the result if we use error tolerant rational reconstruction as in Algorithm 5.

To simplify our presentation in what follows, we will systematically ignore the primes discussed in Remark 5.1. We suppose that we are given a construction which associates to I a uniquely determined ideal $U(0) \subseteq W$, and to each reduction I_p, with p a prime number, a uniquely determined ideal $U(p) \subseteq W_p$, where we make the following assumption:

ASSUMPTION: *We ask that $U(0)_p = U(p)$ for all but finitely many p.*

We write $G(0)$ for the uniquely determined reduced Gröbner basis of $U(0)$, and $G(p)$ for that of $U(p)$. In the applications we have in mind, we wish to construct the unknown ideal $U(0)$ from a collection of its characteristic p counterparts $U(p)$. Technically, given a finite set of primes $\mathcal{P}$, we wish to construct $G(0)$ by computing the $G(p)$, $p \in \mathcal{P}$, and lifting the $G(p)$ coefficientwise to characteristic zero. Here, to identify Gröbner basis elements corresponding to each other, we require that $\mathrm{LM}(G(p)) = \mathrm{LM}(G(q))$ for all $p, q \in \mathcal{P}$. This leads to condition (1b) below:

Definition 5.2. With notation as above, we define:

(1) A prime number p is called *lucky* if the following hold:
   (a) $U(0)_p = U(p)$ and
   (b) $\mathrm{LM}(G(0)) = \mathrm{LM}(G(p))$.
   Otherwise p is called *unlucky*.
(2) If $\mathcal{P}$ is a finite set of primes, set

$$N' = \prod_{p \in \mathcal{P} \text{ lucky}} p \quad \text{and} \quad M = \prod_{p \in \mathcal{P} \text{ unlucky}} p.$$

[1]Note that rescaling to integer coefficients is not helpful: reducing the rescaled generators may yield the wrong leading ideal. See Remark 5.3.

Then $\mathcal{P}$ is called *sufficiently large* if

$$N' > (a^2 + b^2) \cdot M$$

for all coefficients $\frac{a}{b}$ of polynomials in $G(0)$ (assume $\gcd(a, b) = 1$).

Note that a prime p violating condition (1a) is of type-5, while (1b) is a type-4 condition.

Remark 5.3. A modular algorithm for the fundamental task of computing Gröbner bases is presented in [2] and [8]. In contrast to our situation here, where we wish to find the ideal $U(0)$ by computing its reduced Gröbner basis $G(0)$, Arnold's algorithm starts from an ideal which is already given. If p is a prime number, $J \subseteq W$ is an ideal, $H(0)$ is the reduced Gröbner basis of J, and $H(p)$ is the reduced Gröbner basis of J_p, then p is lucky for J in the sense of Arnold if $\mathrm{LM}(H(0)) = \mathrm{LM}(H(p))$. It is shown in [2, Thm. 5.12 and 6.2] that if J is homogeneous and p is lucky for J in this sense, then $H(0)_p$ is well defined and equal to $H(p)$. Furthermore, by [2, Cor. 5.4 and Thm. 5.13], all but finitely many primes are Arnold-lucky for a homogeneous J. Using weighted homogenization as in the proof of [8, Thm. 2.4], one shows that these results also hold true in the nonhomogeneous setup.

Example 5.4. Consider the ideal $J = \langle \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \rangle$, where $f = x^5 + y^{11} + xy^9 + x^3y^9$. The leading terms of the lexicographical Gröbner basis with integral coprime coefficients are as follows:

$$264627y^{39} + \ldots,$$

$$12103947791971846719838321886393392913750065060875xy^8 - \ldots,$$

$$40754032969602177507873137664624218564815033875x^4 + \ldots.$$

Hence, the Arnold unlucky primes for J are

$$3, 5, 11, 809, 65179, 531264751, 43105193484678662861546339 3.$$

With respect to our notion of lucky as introduced in Definition 5.2(1), we first observe:

Lemma 5.5. *The set of unlucky primes is finite.*

Proof. By our general assumption, $U(0)_p = U(p)$ for all but finitely many primes p. Given a prime p such that $U(0)_p = U(p)$, we have $\mathrm{LM}(G(0)) = \mathrm{LM}(G(p))$ if p does not divide the denominator of any coefficient of any polynomial occurring when testing whether $G(0)$ is a Gröbner basis using Buchberger's criterion. The result follows. □

The type-5 condition (1a) cannot be checked a priori: We compute $G(p)$ and, thus, $U(p)$ on our way, but $U(0)_p$ is only known to us after $G(0)$ and, thus, $U(0)$ have been found. However, this is not a problem if we apply error tolerant rational reconstruction since the finitely many bad primes leading to an ideal $U(p)$ different from $U(0)_p$ will not influence the final result:

Lemma 5.6. *If $\mathcal{P}$ is a sufficiently large set of primes satisfying condition (1b), then the reduced Gröbner bases $G(p)$, $p \in \mathcal{P}$, lift via Algorithm 5 to the reduced Gröbner basis $G(0)$.*

Proof. By assumption, a prime $p \in \mathcal{P}$ is lucky if and only if it satisfies condition (1a). In this case, p is Arnold-lucky for $U(0)$. Hence, as pointed out in Remark 5.3, $G(0)_p = G(p)$. Since we assume that $\mathcal{P}$ is sufficiently large, it is then clear from Lemma 4.3 that the coefficients of the Chinese remainder lift $G(N)$, $N = \prod_{p\in\mathcal{P}}$, have a lift to characteristic zero. By the uniqueness statement of Lemma 4.3, this lift coincides with $G(0)$. □

Lemma 5.5 guarantees, in particular, that a sufficiently large set $\mathcal{P}$ of primes satisfying condition (1b) exists. So from a theoretical point of view, the idea of finding $G(0)$ is now as follows: Consider such a set $\mathcal{P}$, compute the reduced Gröbner bases $G(p)$, $p \in \mathcal{P}$, and lift the results to $G(0)$ as described above.

From a practical point of view, we face the problem that we can only test a posteriori whether $\mathcal{P}$ is a sufficiently large set of primes satisfying condition (1b). However, to remedy the situation with respect to (1b), we can proceed in the following randomized way:

First, fix an integer $t \geq 1$ and choose a set of t primes $\mathcal{P}$ at random. Second, compute $\mathcal{GP} = \{G(p) \mid p \in \mathcal{P}\}$, and use a majority vote with respect to the type-4 condition (1b):

DELETEBYMAJORITYVOTE: *Define an equivalence relation on $\mathcal{P}$ by setting $p \sim q :\Longleftrightarrow \mathrm{LM}(G(p)) = \mathrm{LM}(G(q))$. Then replace $\mathcal{P}$ by an equivalence class of largest cardinality*[2]*, and change $\mathcal{GP}$ accordingly.*

Now all Gröbner bases in $\mathcal{GP}$ have the same set of leading monomials. Hence, we can apply Algorithm 5 to the coefficients of the Gröbner bases in $\mathcal{GP}$. If this algorithm returns `false` at some point, we enlarge the set $\mathcal{P}$ by t primes not used so far, and repeat the whole process. Otherwise, the lifting yields a set of polynomials $G \subset R$. Furthermore, if $\mathcal{P}$ is suffciently large, all primes in $\mathcal{P}$ satisfy (1b). However, since we do not know whether $\mathcal{P}$ is sufficiently large, a final verification in characteristic zero is needed. As this may be expensive, especially if $G \neq G(0)$, we first perform a test in positive characteristic:

PTEST: *Randomly choose a prime number $p \notin \mathcal{P}$ such that p does not divide the numerator or denominator of any coefficient occurring in a polynomial in G. Return* `true` *if $G_p = G(p)$, and* `false` *otherwise.*

If PTEST returns `false`, then $\mathcal{P}$ is not sufficiently large (or the extra prime chosen in PTEST is bad). In this case, we enlarge the set $\mathcal{P}$ as above and repeat the process. If PTEST returns `true`, however, then most likely $G = G(0)$. Hence, it makes sense to verify the result over the rationals. If the verification fails, we again enlarge $\mathcal{P}$ and repeat the process.

We summarize this approach in Algorithm 6 (recall that we ignore the primes from Remark 5.1 in our presentation).

Remark 5.7. If Algorithm 6 requires more than one round of the loop, we have to use a weighted cardinality count in DELETEBYMAJORITYVOTE: when enlarging $\mathcal{P}$, the total weight of the elements already present must be strictly smaller than the total weight of the new elements. Otherwise, though highly unlikely in practical terms, it may happen that only unlucky primes are accumulated.

[2]When computing the cardinality, take Remark 5.7 into account.

Algorithm 6. Reconstruction of an Ideal

Input: An algorithm to compute $G(p)$ from I_p, for each prime p, and a way of verifying that a given Gröbner basis over $\mathbb{Q}$ equals $G(0)$.
Output: The Gröbner basis $G(0)$.

choose a list $\mathcal{P}$ of random primes
$\mathcal{GP} = \emptyset$
loop
  for $p \in \mathcal{P}$ **do**
    compute $G(p) \subseteq W_p$
    $\mathcal{GP} = \mathcal{GP} \cup \{G(p)\}$
  $(\mathcal{P}, \mathcal{GP}) =$ DELETEBYMAJORITYVOTE$(\mathcal{P}, \mathcal{GP})$
  lift $\mathcal{GP}$ to $G \subseteq W$ via Chinese remaindering and Algorithm 5
  if the lifting succeeds and PTEST$(I, G, \mathcal{P})$ **then**
    if $G = G(0)$ **then**
      return G
  enlarge $\mathcal{P}$ with primes not used so far

Remark 5.8. Our lifting process works since reduced Gröbner bases are uniquely determined. In practical terms, however, there is often no need to reduce the Gröbner bases involved: it is only required that the construction associating the Gröbner bases to I and its reductions yields uniquely determined results.

Remark 5.9. We may allow that the given construction does not work for finitely many primes p (which are then bad of type-2). In this case, the respective primes will be rejected.

Remark 5.10. Depending on the particular implementation of the construction, type-3 primes (in addition to those considered in Remark 5.1) may occur. In such a situation, it is often cheaper to rely on error tolerance rather than spending computation time to detect these primes.

Example 5.11. If K is any field, and $I \subseteq K[X]$ is a prime ideal, the *normalization* $\overline{A}$ of the domain $A = K[X]/I$ is the integral closure of A in its field of fractions $\mathrm{Q}(A)$. If K is perfect, the normalization algorithm given in [7] will find a "valid denominator" $d \in A$ and an ideal $U \subseteq A$ such that $\frac{1}{d}U = \overline{A} \subseteq \mathrm{Q}(A)$. In fact, U is uniquely determined if we fix d. In practical terms, d and U are a polynomial and an ideal in $K[X]$, respectively. If $K = \mathbb{Q}$, we can apply the modular version of the algorithm (see [3]). This version relies on choosing a "universal denominator" d which is used over the rationals as well as in finite characteristic. Given a prime number p, it may then happen that I_p is not a prime ideal (a type-2 condition), that the leading ideal of I_p does not coincide with that of I (a type-3 condition), that d_p is not defined (a type-1 condition), or that d_p is not a valid denominator (a type-2 condition). In accordance with the general setup, the numerator ideal $U(p)$ is obtained by computing the Gröbner basis $G(p)$, and $\mathrm{LM}(G(0)) = \mathrm{LM}(G(p))$ and $U(0)_p = U(p)$ are type-4 and type-5 conditions, respectively.

The normalization algorithm mentioned above in Example 5.11 finds $\overline{A}$ by successively enlarging A in form of an endomorphism ring of a so-called test ideal $J \subseteq A$. For practical purposes, the radical of the Jacobian ideal is used for J. The following example shows that, for the algorithm computing the radical of the

Jacobian, bad primes p exist which satisfy the type-4 condition (1b) but violate the type-5 condition (1a) in Definition 5.2. In particular, these primes cannot be eliminated by a majority vote on the leading ideal.

Example 5.12. We construct a sextic curve $C = V(I) \subset \mathbb{P}^2_{\mathbb{C}}$, given by an ideal $I = \langle f \rangle \subset \mathbb{Q}[x, y, z]$, such that 5 is a bad prime of type-5 for computing the radical of the singular locus of C. The basic idea is to construct a curve which has two double points in characteristic 0, which coincide when reducing modulo 5, while one additional double point appears.

For

$$L = \langle y,\ x - 4z \rangle^2 \cap \langle y,\ x + 6z \rangle^2 \subseteq \mathbb{Q}[x, y, z],$$

the reduced Gröbner basis with respect to the degree reverse lexicographical ordering is

$$G = \{G_1, G_2, G_3\} = \left\{ y^2,\ (x - 4z)(x + 6z)y,\ (x - 4z)^2(x + 6z)^2 \right\}.$$

Note, that both L and

$$L_5 = \left\langle y^2,\ (x + z)^2 y,\ (x + z)^4 \right\rangle \subseteq \mathbb{F}_5[x, y, z]$$

have the same leading semigroup $\left\langle y^2, x^2 y, x^4 \right\rangle$.

Writing generators of

$$M = L_5 \cap \langle y,\ x - z \rangle^2 \subset L_5$$

in terms of the Gröbner basis of L_5 yields the representation

$$M = \left\langle y^2,\ (x - z) \cdot (x + z)^2 y,\ (x^2 + 3xz + z^2) \cdot (x + z)^4 \right\rangle.$$

Now consider a generic homogeneous element of degree 6 in

$$\left\langle G_1,\ (x - z) \cdot G_2,\ (x^2 + 3xz + z^2) \cdot G_3 \right\rangle \subset \mathbb{Q}[x, y, z].$$

For practical purposes, a random element will do, for example,

$$f = x^6 + y^6 + 7x^5 z + x^3 y^2 z - 31x^4 z^2 - 224x^3 z^3 + 244x^2 z^4 + 1632xz^5 + 576z^6.$$

For the ideal $I = \langle f \rangle$, the prime 5 is bad of type-5 with respect to the algorithm

$$I \mapsto \sqrt{I + \mathrm{Jac}(I)},$$

where $\mathrm{Jac}(I)$ denotes the Jacobian ideal of I: First note, that no coefficient of f is divisible by 5. In particular, $\mathrm{LM}(I) = \left\langle x^6 \right\rangle = \mathrm{LM}(I_5)$, so 5 is Arnold-lucky for I. We compute

$$U(0) = \sqrt{I + \mathrm{Jac}(I)} = \langle y, x - 4z \rangle \cap \langle y, x + 6z \rangle,$$
$$U(5) = \sqrt{I_5 + \mathrm{Jac}(I_5)} = \left\langle y, x^2 - z^2 \right\rangle = \langle y, x - z \rangle \cap \langle y,\ x + z \rangle,$$

and

$$U(0)_5 = \left\langle y, (x + z)^2 \right\rangle.$$

Hence

$$\mathrm{LM}(U(0)) = \left\langle y, x^2 \right\rangle = \mathrm{LM}(U(5)),$$

but

$$U(0)_5 \neq U(5).$$

Acknowledgement

The authors would like to thank the referees who made valuable suggestions which improved the presentation of this paper.

References

[1] W. W. Adams and P. Loustaunau, *An Introduction to Gröbner Bases*, Graduate Studies in Mathematics, vol. 3, American Mathematical Society, Providence, RI, 1994. MR1287608 (95g:13025)

[2] E. A. Arnold, *Modular algorithms for computing Gröbner bases*, J. Symbolic Comput. **35** (2003), no. 4, 403–419, DOI 10.1016/S0747-7171(02)00140-2. MR1976575 (2004c:13044)

[3] J. Böhm, W. Decker, S. Laplagne, G. Pfister, A. Steenpaß, and S. Steidel, *Parallel algorithms for normalization*, J. Symbolic Comput. **51** (2013), 99–114, DOI 10.1016/j.jsc.2012.07.002. MR3005784

[4] J. Böhm, W. Decker, S. Laplagne, and G. Pfister, *Local to global algorithms for the Gorenstein adjoint ideal of a curve*. In preparation.

[5] G. E. Collins and M. J. Encarnación, *Efficient rational number reconstruction*, J. Symbolic Comput. **20** (1995), no. 3, 287–297, DOI 10.1006/jsco.1995.1051. MR1378101 (97c:11116)

[6] M. J. Encarnación, *Computing GCDs of polynomials over algebraic number fields*, J. Symbolic Comput. **20** (1995), no. 3, 299–313, DOI 10.1006/jsco.1995.1052. MR1378102 (97c:11117)

[7] G.-M. Greuel, S. Laplagne, and F. Seelisch, *Normalization of rings*, J. Symbolic Comput. **45** (2010), no. 9, 887–901, DOI 10.1016/j.jsc.2010.04.002. MR2661161 (2011m:13012)

[8] N. Idrees, G. Pfister, and S. Steidel, *Parallelization of modular algorithms*, J. Symbolic Comput. **46** (2011), no. 6, 672–684, DOI 10.1016/j.jsc.2011.01.003. MR2781946 (2012b:68331)

[9] P. Q. Nguyen and D. Stehlé, *Low-dimensional lattice basis reduction revisited*, ACM Trans. Algorithms **5** (2009), no. 4, Art. 46, 48, DOI 10.1145/1597036.1597050. MR2571909 (2011c:68236)

[10] P. Kornerup and R. T. Gregory, *Mapping integers and Hensel codes onto Farey fractions*, BIT **23** (1983), no. 1, 9–20, DOI 10.1007/BF01937322. MR689601 (84b:10015)

[11] P. S. Wang, *A p–adic algorithm for univariate partial fractions*. Proceedings SYMSAC '81, 212–217 (1981).

[12] P. S. Wang, M. J. T. Guy, and J. H. Davenport, *P–adic reconstruction of rational numbers*. SIGSAM Bull, 2–3 (1982).

Fachbereich Mathematik, Technical University Kaiserslautern, Postfach 3049, 67653 Kaiserslautern, Germany

E-mail address: boehm@mathematik.uni-kl.de

Fachbereich Mathematik, Technical University Kaiserslautern, Postfach 3049, 67653 Kaiserslautern, Germany

E-mail address: decker@mathematik.uni-kl.de

Fachbereich Mathematik, Technical University Kaiserslautern, Postfach 3049, 67653 Kaiserslautern, Germany

E-mail address: fieker@mathematik.uni-kl.de

Fachbereich Mathematik, Technical University Kaiserslautern, Postfach 3049, 67653 Kaiserslautern, Germany

E-mail address: pfister@mathematik.uni-kl.de