Computing and Using Minimal Polynomials

John Abbott

Institut für Mathematik, Universität Kassel, Germany

Anna Maria Bigatti

Dip. di Matematica, Università degli Studi di Genova, Via Dodecaneso 35, I-16146 Genova, Italy

Elisa Palezzato

Dip. di Matematica, Università degli Studi di Genova, Via Dodecaneso 35, I-16146 Genova, Italy

Lorenzo Robbiano

Dip. di Matematica, Università degli Studi di Genova, Via Dodecaneso 35, I-16146 Genova, Italy

Abstract

Given a zero-dimensional ideal I in a polynomial ring, many computations start by finding univariate polynomials in I. Searching for a univariate polynomial in I is a particular case of considering the minimal polynomial of an element in P/I. It is well known that minimal polynomials may be computed via elimination, therefore this is considered to be a "resolved problem". But being the key of so many computations, it is worth investigating its meaning, its optimization, its applications (*e.g.* testing if a zero-dimensional ideal is radical, primary or maximal). We present efficient algorithms for computing the minimal polynomial of an element of P/I. For the specific case where the coefficients are in $\mathbb{Q}$, we show how to use modular methods to obtain a guaranteed result. We also present some applications of minimal polynomials, namely algorithms for computing radicals and primary decompositions of zero-dimensional ideals, and also for testing radicality and maximality.

Key words: Minimal polynomial, Gröbner bases, modular methods, radical, maximal, primary
1991 MSC: [2010] 13P25, 13P10, 13-04, 14Q10, 68W30

Email addresses: `abbott@dima.unige.it` (John Abbott), `bigatti@dima.unige.it` (Anna Maria Bigatti), `bigatti@dima.unige.it` (Elisa Palezzato), `robbiano@dima.unige.it` (Lorenzo Robbiano).

[1] This research was partly supported by the project H2020-FETOPN-2015-CSA_712689 of the Euro-

1. Introduction

This paper describes both theoretical and practical aspects of computing minimal polyomials, with particular emphasis on our practical implementation in CoCoALib (Abbott and Bigatti, 2019) and CoCoA (Abbott et al., 2019). It is organized into three parts: computing minimal polynomials over $\mathbb{F}_p$, computing minimal polynomials over $\mathbb{Q}$, and using minimal polynomials.

In linear algebra it is frequently necessary to use non-linear objects such as minimal and characteristic polynomials since they capture fundamental information about endomorphisms of finite-dimensional vector spaces. It is well-known that if K is a field and R is a zero-dimensional affine K-algebra (*i.e.* a zero-dimensional algebra of the form $R = K[x_1, \dots, x_n]/I$) then R is a finite-dimensional K-vector space — *e.g.* see Proposition 3.7.1 of (Kreuzer and Robbiano, 2008). Consequently, it is not surprising that minimal and characteristic polynomials can be successfully used to detect properties of R.

This point of view was taken systematically in the book by Kreuzer and Robbiano (2016), where the particular importance of minimal polynomials (rather greater than that of characteristic polynomials) emerged quite clearly. The book clarified one main advantage of minimal polynomials over characteristic polynomials, namely that minimal polynomials generalise to families of pairwise commuting endomorphisms, while characteristic polynomials do not. The book also described several algorithms which use minimal polynomials as a crucial tool. The approach taken there was a good source of inspiration for our research, so we decided to delve into the theory of minimal polynomials, their uses, and their applications.

Being such fundamental tools in linear algebra, minimal polynomials have played a prominent role in several branches of research over the last two centuries. As a consequence, it is impractical to track down all contributions to this theory. Two further sources of particular inspiration were the paper by Lazard (1992), where minimal polynomials were systematically used in the process of solving zero-dimensional systems of polynomial equations, and the paper by Bostan et al. (2003), which contains a fine analysis of the complexity of computing minimal polynomials.

Relevance to the SC^2 *community*

It is not immediately apparent how minimal polynomials could be relevant to the SC^2 community. However, our methods for computing them efficiently are very helpful for solving polynomial systems, and are basic building blocks also for a number of other applications in the SC^2 community, as introduced in (Abbott and Bigatti, 2017).

For instance, one fundamental tool in the algebraic approach to tackling SC^2 problems is *Cylindrical Algebraic Decomposition*, often abbreviated to *CAD* (first introduced in (Collins, 1975)). The software CArL/SMT-RAT by Kremer and Ábrahám (2018) employs Lazard's variant of CAD (McCallum et al., 2017) which requires polynomial factorization over algebraic field extensions. We use our efficient algorithms for minimal

pean Union.   J. Abbott is an INdAM-COFUND Marie Curie Fellow.

Published on JSC https://doi.org/10.1016/j.jsc.2019.07.022

polynomials to compute quickly the primary decomposition of zero-dimensional ideals (see Section 4.5 and Table 5), which in turn give a good method for factorizing polynomials over algebraic field extensions. We presented the relevant theory and our CoCoALib implementation in (Abbott et al., 2018a); further background material is in Section 5.3.B of (Kreuzer and Robbiano, 2016), Indeed, our CoCoALib implementation is used by CArL/SMT-RAT.

Fundamentals of our approach

The first step of our approach was to devise and implement good algorithms for computing the minimal polynomial of an element of R and that of a K-endomorphism of R (see Algorithms 2.8, and 2.10). They are described in Section 2, and refine similar algorithms examined in the book by Kreuzer and Robbiano (2016). They have been implemented in CoCoALib (Abbott and Bigatti, 2019), and are accessible from CoCoA (Abbott et al., 2019), as indeed are all other algorithms described in this paper. Although the theoretical content of Section 2 is essentially elementary, many remarks on implementation details are given to get the good performance shown in Table 1.

Efficient computation by modular techniques

Section 3 constitutes a contribution of great practical significance: it addresses the problem of computing minimal polynomials of elements of an affine $\mathbb{Q}$-algebra using a modular approach. The technique of using modular reduction has a long tradition, see for instance these articles by Pauer (2007), Winkler (1988), Gräbe (1988), Noro and Yokoyama (1999), Noro and Yokoyama (2004), Noro and Yokoyama (2018), Aoyama and Noro (2018), Noro (2002), Idrees et al. (2011) and Monagan (2004). Usually modular methods are associated with results which are correct *only with high probability.* In contrast, with the approach we introduce here, the minimal polynomial is guaranteed to be correct (see Remark 3.23).

Why might a modular approach be useful for computing minimal polynomials? Let us have a look at an example. Let $\mathbb{X} = \{p_1, p_2, p_3, p_4\}$ be the set of the following four points in $\mathbb{A}^3_{\mathbb{Q}}$: $p_1 = (1, 3, 0)$, $p_2 = (1, 0, 4)$, $p_3 = (-5, 7, 1)$, $p_4 = (1, 0, 0)$. The vanishing ideal of $\mathbb{X}$ in the polynomial ring $P = \mathbb{Q}[x, y, z]$ is $I = \langle z^2-\frac{1}{2}x-4z+\frac{1}{2}, yz+\frac{7}{6}x-\frac{7}{6},\ y^2+\frac{14}{3}x-3y-\frac{14}{3}\rangle$. We take $f = 2x^2 + 3y^4 + 5z^6 \in P$. Then the minimal polynomial of $\bar{f}$ in P/I, *i.e.* the lowest-degree, monic polynomial $g(z)$ such that $g(f) \in I$, is $z^4-27987\,z^3+155510626\,z^2-36732206532\,z + 72842594440$. Even in this "small" example, although both the ideal and the element f have simple coefficients, the minimal polynomial has much larger coefficients.

The modular approach tames the complexity of computing the big coefficients of such polynomials. However, as always happens with a modular approach, various obstacles have to be overcome — see for instance the discussion contained in (Abbott et al., 2017). In particular, we deal with the notion of reduction of an ideal modulo p, and we do so by introducing the notion of σ-denominator of an ideal (see Definition 3.6 and Theorem 3.7), which enables us to surmount these obstacles. The reason why we introduce the σ-denominator is that, for a given a term ordering, the reduced Gröbner basis of an ideal is unique, so that the theoretical results do not depend on the generators of the ideal, but just on the ideal itself.

Hand-in-hand with the modular approach go the notions of *usable, good* and *bad* primes (see Definitions 3.17 and 3.18). We show that all but finitely many primes are good (see Theorem 3.20 and Corollary 3.21), and this paves the way to the construction of the fundamental Algorithm 3.22. In the literature, several authors have looked at various notions of bad reduction in similar contexts, see for instance the articles by Noro and Yokoyama (2018), Pauer (2007), Winkler (1988) and Arnold (2003). However, our approach is systematically tied to reduced Gröbner bases as computationally robust representations of ideals; we study more deeply this approach in the preprint (Abbott et al., 2018b). The combination of the theoretical results explained in this section with various implementation details lead to the good practical performance as shown in Table 2. For example, the choice of larger "small primes" as moduli turned out to be a winning strategy, namely with about 30 bits (on a 64-bit platform).

Uses of minimal polynomials: radical, primary decomposition

Section 4 shows how minimal polynomials can be successfully and efficiently used to compute several important invariants of zero-dimensional affine K-algebras. More specifically, in Subsection 4.1 we describe Algorithms 4.7 and 4.8 which show respectively how to determine whether a zero-dimensional ideal is radical, and how to compute the radical of a zero-dimensional ideal. In Subsection 4.2 we present several algorithms which determine whether a zero-dimensional ideal is maximal or primary. The techniques used depend very much on the field K. The main distinction is between small finite fields and fields of characteristic zero or big fields of positive characteristic. In particular, it is noteworthy that in the first case Frobenius spaces play a fundamental role — *e.g.* see Section 5.2 of the book by Kreuzer and Robbiano (2016).

Finally, in Subsection 4.5 a series of algorithms (see Algorithms 4.24, 4.28, 4.29 and 4.30) describe how to compute the primary decomposition of a zero-dimensional affine K-algebra. They are inspired by the content of Chapter 5 of (Kreuzer and Robbiano, 2016), but also present many novelties.

Implementation and timings

As already mentioned, all the algorithms described in this paper have been implemented in CoCoA. Their merits are illustrated by the tables of examples contained in Sections 2 and 3, and also at the end of Section 4. The examples were chosen to cover a wide spectrum of zero-dimensional affine K-algebras; some are complete intersections, and some are not. The experiments were performed on a MacBook Pro with Intel Core i7 processor (clocked at 2.9GHz), using our implementation in CoCoA 5.

At the request of the referees we supply comparative timings against two (free and open-source) competitors, despite our reservations. But is our comparison really fair? Both Singular (by Decker et al. (2019)) and Macaulay2 (by Grayson and Stillman (2019)) are advanced software systems which require months to learn thoroughly; for the comparison, we consulted their respective manuals, and chose what appear to be the most relevant functions.

We tried to compute our examples with Macaulay2, but were unable to compute them in a reasonable time. In contrast, we could compute most of the examples with Singular. The results are in Table 5, and confirm the efficiency of our algorithms and implementations (whose outputs, as already mentioned, are guaranteed to be correct).

2. Computing Minimal Polynomials

Here we introduce the notation and terminology we shall use, and the definition of minimal polynomial which is the fundamental object studied in this paper.

Let K be a field, let $P = K[x_1, \dots, x_n]$ be a polynomial ring in n indeterminates, and let $\mathbb{T}^n$ denote the monoid of power-products in $x_1, \dots, x_n$. Let I be a zero-dimensional ideal in P; this implies that the ring $R = P/I$ is a zero-dimensional affine K-algebra, hence it is a finite dimensional K-vector space. Then, for any f in P there is a linear dependency mod I among the powers of f: in other words, there is a polynomial $g(z) = \sum_{i=0}^{d} \lambda_i z^i \in K[z]$ which vanishes modulo I when evaluated at $z = f$.

Example 2.1. Let $P = \mathbb{F}_{101}[x, y]$, $I = \langle x^2 - y,\ y^2 - 2x - 4\rangle$, and $f = 5x - 3y$. Then $g(z) = z^4 + 18z^2 + 48z - 23 \in K[z]$ is such that $g(f) = f^4 + 18f^2 + 48f - 23 \in I$, or equivalently $g(\bar{f}) = 0 \in P/I$. Moreover $g(z)$ is the lowest degree monic polynomial with this property.

Definition 2.2. Let K be a field, let $P = K[x_1, \dots, x_n]$, and let I be a zero-dimensional ideal. Given a polynomial $f \in P$, we have a K-algebra homomorphism $K[z] \to P/I$ given by $z \mapsto f \bmod I$. The monic generator of the kernel of this homomorphism is called the **minimal polynomial** of $f \bmod I$ (or simply "of f" when the ideal I is obvious), and is denoted by $\mu_{f,I}(z)$.

Remark 2.3. The particular case of $\mu_{x_i,I}(x_i)$, where x_i is an indeterminate, is a very important and popular object when computing: in fact $\mu_{x_i,I}(x_i)$ is the lowest degree polynomial in x_i belonging to I, that is $I \cap K[x_i] = \langle \mu_{x_i,I}(x_i)\rangle$.

Example 2.4. Let $P = \mathbb{F}_{101}[x, y]$, $I = \langle y^3 - xy - 2y^2 + y,\ xy^2,\ x^2 - x\rangle$, and $f = y$. Then $\mu_{f,I}(y) = y^4 - 2y^3 + y^2 = y^2 \cdot (y-1)^2$.

Example 2.5. Let $P = \mathbb{Q}[x, y]$, $I = \langle x^2 - \frac{1}{7}y^2 - 5,\ y^2 + 4x - \frac{7}{2}\rangle$, and $f = 3x - 2y$. Then $\mu_{f,I}(z) = z^4 + \frac{24}{7}z^3 - \frac{6527}{49}z^2 + \frac{5868}{7}z + \frac{10967}{28}$. We will see in Example 3.24 how we compute it with a modular approach.

For the basic properties of Gröbner bases we refer to Buchberger (1985) and Kreuzer and Robbiano (2008). Let σ be a term-ordering on $\mathbb{T}^n$, and let I be an ideal in the polynomial ring P. It is known that $\mathrm{NF}_{\sigma,I}(f)$, the σ-normal form of f with respect to I, does not depend on which σ-Gröbner basis of I is used nor on which specific rewriting steps were used to calculate it (see Proposition 2.4.7 in Kreuzer and Robbiano (2008)). If I is clear from the context, we write simply $\mathrm{NF}_\sigma(f)$.

Remark 2.6 (Elimination)**.** A well-known method for computing the minimal polynomial $\mu_{f,I}(z)$ is by elimination. One extends P with a new indeterminate to produce a new polynomial ring $R = K[x_1, \dots, x_n,\ z]$, then defines the ideal $J = IR + \langle z - f\rangle$ in R, and finally eliminates the indeterminates $x_1, \dots, x_n$.

However, even in the case where f is just an indeterminate, the algorithm which derives from this approach is usually impractically slow on non-trivial examples; though the performance could be improved by using FGLM (see Remark 2.13).

Another way to compute $\mu_{f,I}(z)$ is via multiplication endomorphisms on P/I. Let $f \in P$ then we write $\vartheta_{\bar{f}} : P/I \to P/I$ for the endomorphism "multiplication by $\bar{f}$". There is a natural isomorphism between P/I and $K[\vartheta_{\bar{f}} \mid \bar{f} \in P/I]$ which associates $\bar{f}$ with $\vartheta_{\bar{f}}$ (see Proposition 4.1.2 in Kreuzer and Robbiano (2016)).

Example 2.7 (Example 2.4 continued)**.** Let $P = \mathbb{F}_{101}[x, y]$, $I = \langle y^3 - xy - 2y^2 + y,\ xy^2,\ x^2 - x\rangle$ and $f = y$. The given generators are a DegRevLex-Gröbner basis, thus a quotient basis for P/I is $QB = \{1, y, y^2, x, xy\}$, and the matrix for "multiplication by $\bar{f}$" with respect to the basis QB is $A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$. In CoCoA this can be computed via the call `MultiplicationMat(y, I, QB)`. Note that the matrix is computed by calculating $\mathrm{NF}_I(f \cdot t)$ for all $t \in QB$ (see Remark 2.11).

The minimal polynomial of f modulo I is the same as the minimal polynomial of the endomorphism $\vartheta_{\bar{f}}$. Thus, if the matrix A represents $\vartheta_{\bar{f}}$ with respect to some K-basis of P/I, we can compute the minimal polynomial of A (and thus of $\vartheta_{\bar{f}}$) using the following algorithm which is a refined version of Algorithm 1.1.8 in Kreuzer and Robbiano (2016).

```
Algorithm 2.8. MinPolyQuotMat
notation: Let $P = K[x_1, \dots, x_n]$ with term-ordering $\sigma$
Input $I$, a zero-dimensional ideal in $P$, and a polynomial $f \in P$
1 compute $GB$, a $\sigma$-Gröbner basis for $I$;
    from $GB$ compute $QB$, the corresponding monomial quotient basis of $P/I$
    (below we assume 1 is the first element in $QB$)
2 compute $A$, the matrix representing the map $\vartheta_{\bar{f}}$ w.r.t. $QB$
3 let $v_0 = (1\ 0\ 0\ \dots)^{\mathrm{tr}}$ and $L = \{v_0\}$
4 Main Loop: for $i = 1, 2, \dots, \mathrm{len}(QB)$ do
    4.1 let $v_i = A \cdot v_{i-1}$     (hence we have $v_i = A^i \cdot v_0$)
    4.2 is there a linear dependency $v_i = \sum_{j=0}^{i-1} c_j v_j$ with coefficients $c_j \in K$?
        4.2-yes return $\mu_{f,I}(z) = z^i - \sum_{j=0}^{i-1} c_j z^j$
        4.2-no append $v_i$ to $L$
Output $\mu_{f,I}(z) \in K[z]$
```

Example 2.9 (Example 2.7 continued)**.** For $i = 0, 1, \dots$ the vector of v_i comprises the coefficients of $\bar{f}^i$ with respect to the vector space basis (QB in the algorithm). Writing these vectors as columns gives $C = \begin{matrix} 1 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 1 & 0 & -1 & -2 & -3 & \dots \\ 0 & 0 & 1 & 2 & 3 & 4 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 1 & 2 & 3 & \dots \end{matrix}$. Note that the first 4 columns are linearly independent, whereas the first 5 admit the relation $\bar{f}^4 = 2\bar{f}^3 - \bar{f}^2$, hence the algorithm stops at the fourth iteration returning $y^4 - 2y^3 + y^2$. Notice that in this instance the minimal polynomial has degree $< \deg_K(P/I)$, so it strictly divides the characteristic polynomial.

There is a still more direct approach. It comes from considering the very definition of minimal polynomial: we look for the first linear dependency among the powers $\bar{f}^i$ in P/I. Here we give a refined version of Algorithm 5.1.2 in Kreuzer and Robbiano (2016).

```
Algorithm 2.10. MinPolyQuotDef
notation: Let $P = K[x_1, \dots, x_n]$ with term-ordering $\sigma$
Input  $I$, a zero-dimensional ideal in $P$, and a polynomial $f \in P$
1 compute $GB$, a $\sigma$-Gröbner basis for $I$;
    from $GB$ compute $QB$, the corresponding monomial quotient basis of $P/I$
2 let $f = \mathrm{NF}_{\sigma,I}(f)$
3 let $r_0 = f^0$ $(= 1)$ and $L = \{r_0\}$
4 Main Loop: for $i = 1, 2, \dots, \mathrm{len}(QB)$ do
    4.1 compute $r_i = \mathrm{NF}_{\sigma,I}(f \cdot r_{i-1})$    (hence we have $r_i = \mathrm{NF}_{\sigma,I}(f^i)$)
    4.2 is there a linear dependency $r_i = \sum_{j=0}^{i-1} c_j r_j$ with coefficients $c_j \in K$?
        4.2-yes return $\mu_{f,I}(z) = z^i - \sum_{j=0}^{i-1} c_j z^j$
        4.2-no append $r_i$ to $L$
Output  $\mu_{f,I}(z) \in K[z]$
```

Remark 2.11 (MinPolyQuotMat vs. MinPolyQuotDef)**.** Notice that algorithms MinPolyQuotMat and MinPolyQuotDef essentially do the same computation: the first using a matrix representation, and the second a polynomial representation. The main intrinsic difference is that the first algorithm computes the normal forms when constructing the multiplication matrix, whereas the second computes the normal forms in the main loop.

Example 2.12 (Example 2.9 continued)**.** In this algorithm we do not compute the multiplication matrix, but we compute the normal forms of successive powers of f: hence $r_0 = 1$, $r_1 = \mathrm{NF}_I(f) = y$, $r_2 = \mathrm{NF}_I(f \cdot r_1) = y^2$, $r_3 = \mathrm{NF}_I(f \cdot r_2) = xy + 2y^2 - y$, $r_4 = \mathrm{NF}_I(f \cdot r_3) = 2xy + 3y^2 - 2y$, and so on. From these we implicitly construct the same sequence of columns C, and thence obtain the same relation, $\mu_{f,I}(y) = y^4 - 2y^3 + y^2$.

Remark 2.13 (MinPolyQuotMat/Def vs. FGLM-Elimination)**.** The performance of the naive elimination algorithm (Remark 2.6) could be greatly improved using FGLM (see Faugère et al. (1993)). Given a Gröbner basis of a zero-dimensional ideal, we can compute an elimination Gröbner basis using linear algebra on the quotient basis. Note that computing a Gröbner basis with respect to an elimination ordering produces the desired minimal polynomial along with many other “superfluous” polynomials (which are necessary to complete the Gröbner basis, and which typically have “uglier” coefficients than the minimal polynomial). We can modify the FGLM–elimination process to stop the Gröbner basis computation as soon as the minimal polynomial is found. The result would then be effectively quite similar to the algorithms we presented above.

These two algorithms, MinPolyQuotMat and MinPolyQuotDef, are indeed quite simple and natural, but we want to emphasize that a careful implementation is essential for making them efficient. The reward is performance which is dramatically better than the naive elimination approach (see the timings in Subsection 2.1).

There are two crucial steps for achieving an efficient implementation.

Remark 2.14 (Linear Algebra)**.** The common step in both algorithms is the search for a linear dependency (in steps MinPolyQuotMat-4.2 and MinPolyQuotDef-4.2). We implemented it *incrementally* in CoCoALib (Abbott and Bigatti, 2019) creating a C++ object called `LinDepMill`. This object accepts vectors one at a time, and says whether the last vector it was given is linearly dependent on the earlier vectors; if so, then it makes available the representation of the last vector as a linear combination of the earlier vectors. Internally, as new vectors are supplied, `LinDepMill` simply builds up and stores a row-reduced matrix, and keeps track of the corresponding linear representations in terms of the input row-vectors. Moreover, it is easy to implement this class in an efficient way over a (small prime) finite field; in CoCoALib its core is the class `LinDepFp` which uses machine integers. In our experiments, checking for the linear dependency now represents 1 to 3% of the total computation time.

In contrast, the analogous check for the linear dependency over $\mathbb{Q}$ is intrinsically more expensive, and represents most of the total time; this motivates our modular approach presented in Section 3.

Remark 2.15 (Powers and normal forms)**.** When computing over $\mathbb{F}_p$ the most expensive parts of the algorithms is in the computation of the powers, and of the normal forms.

In step MinPolyQuotMat-4.1 and in step MinPolyQuotDef-4.1 we adopt an incremental approach, and do not compute the powers A^i and f^i: this is quite a simple idea, but very important.

Having done that, and considering the computations over $\mathbb{F}_p$, the most expensive operation for MinPolyQuotDef is the normal form (in step 4.1), generally taking more than 95% of the total time. On the other hand, for MinPolyQuotMat the most expensive operation is the multiplication (in step 4.1), generally taking about 90–95% of the time. The construction of the multiplication matrix in MinPolyQuotMat (in step 2) requires several normal forms, but in our implementation, again using an incremental approach, this generally takes less than 10% of the total time.

2.1. Timings: Computing Minimal Polynomials in Finite Characteristic

In this subsection we present some timings for the computation of minimal polynomials of elements in zero-dimensional affine $\mathbb{F}_p$-algebras.

The column **Example** gives the reference number to the examples listed below. The column **GB** gives the times to compute the `DegRevLex`-Gröbner basis (in seconds); the columns **Def, Mat** give the times (in seconds) of the computation of the algorithms 2.10 and 2.8, respectively. We implemented the algorithm MinPolyQuotMat both in dense and in sparse representation (currently called `MinPolyQuotDefLin`). We give the timings for the latter, which is up to twice as fast on the examples below.

Note that CoCoA, whenever computing the Gröbner basis G of an ideal I, stores G within the representation of I. Considering that G is probably precomputed (for example, to check whether I is zero-dimensional) the timings in **Def, Mat** do not include the **GB** time.

The column **deg** gives the degree of the answer, as an indication of the complexity of the output.

As a comparison, we mention that the elimination algorithm in CoCoA takes about 50 and 90 seconds on the first two examples, and more than 10 minutes on all the others (see Remark 2.6).

Table 1. Timings over prime finite fields

Example		**GB**	**dim$_K$**	**MinPoly**		
				deg	**Def**	**Mat**
		time			*time*	*time*
2.16	$f_1 = t$	0.38	501	501	1.86	3.75
	f_2			501	3.15	6.51
	f_3			500	4.49	7.98
2.17	f	0.00	720	720	2.43	12.29
2.18	f	0.17	593	590	4.60	10.02
2.19	f	0.01	464	462	0.90	3.10
2.20	$f_1 = z$	0.00	880	11	0.00	0.02
	f_2			880	1.06	20.68

Example 2.16. The following is an example of a complete intersection in characteristic 101. Let $P = \mathbb{F}_{101}[x, y, z, t]$. Let $g_1 = xyzt - 18z^3 + 16y^2 - 28t^2 - 33x$, $g_2 = xyz^3 - x + y$, $g_3 = x^4 + x^2y - 21y^3 - 7t^2 - 25z$, $g_4 = yt^{11} + 26x^3 + 19z$. Let $I = \langle g_1,\ g_2,\ g_3,\ g_4\rangle$, $f_1 = t$, $f_2 = 3z^4 - 5y + x - t$ and $f_3 = 3y^4z^2 - y^3zt - 12z^4 - y^3 + z^2 - x$.

Example 2.17. This is an example which uses the defining ideal of the splitting algebra of a polynomial of degree 6.

We let $P = \mathbb{F}_{101}[a_1, a_2, a_3, a_4, a_5, a_6]$, and for $j = 1, \ldots, 6$ let s_j be the elementary symmetric polynomial in the indeterminates $a_1, a_2, a_3, a_4, a_5, a_6$. Then the ideal $I = \langle s_1,\ s_2,\ s_3,\ s_4,\ s_5 - 7,\ s_6 - 1\rangle$ is the defining ideal of the splitting algebra of the polynomial $x^6 - 7x + 1$. We let $f = a_1 + 2a_2 + 3a_3 + 4a_4 + 5a_5 + 6a_6$

Example 2.18. This is an example of a complete intersection in "large" characteristic $p = 1000000007$.

Let $P = \mathbb{F}_p[x, y, z, t]$, let $g_1 = x^5yzt + z^3 - y^2 + 73t^2 - 2x$, $g_2 = xyz^6 - x + y$, $g_3 = 2x^4 - x^2y + 34y^3 - 7zt^2$, $g_4 = yt^4 + 26x^3 + z$. Let $I = \langle g_1,\ g_2,\ g_3,\ g_4\rangle$ and $f = x^2t + 5y$.

Example 2.19. This is an example of a non-radical ideal in characteristic $p = 101$ which is not a complete intersection.

Let $P = \mathbb{F}_p[x, y, z]$, let $g_1 = (x^7 - y - 3z)^2$, $g_2 = xy^5 - 7z^2 - 2$, $g_3 = yz^6 - x - z + 14$. Then let $J_1 = \langle g_1, g_2, g_3\rangle$, let $J_2 = \langle x, y, z\rangle^2$, let $I = J_1 \cap J_2$, and let $f = x^2 - 3xy - z$.

Example 2.20. This is an example in characteristic $p = 23$.

Let $P = \mathbb{F}_p[x, y, z]$, let $f_1 = z$, $f_2 = 3x - 2y + 5z$ and $I = \langle g_1, g_2, g_3 \rangle$ where

$$\begin{aligned}
g_1 &= y^5 - 7y^4 + 2y^3 + 11y^2 - y + 5, \\
g_2 &= z^{11} + 9z^{10} - 9z^9 + 7z^8 - 8z^7 - 4z^6 + 9z^5 + z^4 - 5z^3 + 7z^2 + z + 10, \\
g_3 &= x^{16} + 8x^{15} - 6x^{14} - 8x^{13} + 4x^{12} - 4x^{11} + 5x^{10} + 8x^9 + 5x^8 - 4x^7 + \\
&\quad 5x^6 + 2x^5 - 7x^4 + 4x^3 + 10x^2 + 3x + 8
\end{aligned}$$

3. A Modular Approach for Minimal Polynomials

The topic of this section is to show how to compute the minimal polynomial of an element of a zero-dimensional affine $\mathbb{Q}$-algebra using a modular approach. In this section we describe the necessary tools to achieve this goal.

Modular reduction is a very well-known technique, however there is no universal method for addressing the specific problems of bad reduction arising in every application. Our problem is no exception as we shall explain shortly.

Some results in this section are essentially known, for instance Theorem 3.7 is similar to Theorem 1 in Winkler (1988). However, our idea is to stress the theoretical importance of reduced Gröbner bases. The main reason is that, given a term ordering, the reduced Gröbner basis of an ideal is unique, so that the theoretical results depend just on the ideal, and not on the generators given.

In particular, we define the σ-denominator of an ideal (see Definition 3.4) and the (p, σ)-reduction of an ideal (see Definition 3.9). Then we describe the relevant notions of usable, good and bad primes (see Subsection 3.2) and, finally, we put it all together to produce an algorithm which turns out to perform quite well (see Subsection 3.2).

3.1. Reductions of Ideals modulo p

The first matter to address is the following: given an ideal I in $\mathbb{Q}[x_1, \ldots, x_n]$ what does it mean to reduce I modulo a prime number p? Since there is no homomorphism from $\mathbb{Q}$ to $\mathbb{F}_p$, there is no immediate, universal answer to this question. This problem has attracted a lot of attention over many years: various approaches can be found in Winkler (1988), Pauer (2007), Gräbe (1988), Noro and Yokoyama (2018), Arnold (2003), Idrees et al. (2011), Abbott et al. (2017) and Aoyama and Noro (2018). In this section we investigate the problem, and provide a useful answer. We let $P = \mathbb{Q}[x_1, \ldots, x_n]$, and let σ be term-ordering on the power-products in P.

Definition 3.1. Let $\delta \in \mathbb{N}$ be positive, we use the symbol $\mathbb{Z}_\delta$ to denote the **localization of $\mathbb{Z}$ by the multiplicative set generated by** δ, *i.e.* the subring of $\mathbb{Q}$ consisting of numbers represented by fractions of type $\frac{a}{\delta^k}$ where $a \in \mathbb{Z}$ and $k \in \mathbb{N}$. Beware: some authors employ the notation $\mathbb{Z}_p$ to mean the finite field of p elements, or for p-adic numbers. If p is a prime number, we use the symbol $\mathbb{F}_p$ to denote the **finite field** $\mathbb{Z}/p\mathbb{Z}$.

Observe that $\mathbb{Z}_\delta$ depends only on the radical $\mathrm{Rad}(\delta)$, *i.e.* the product of all primes dividing δ. Furthermore, if $\delta_1, \delta_2 \in \mathbb{N}$ are two positive integers then $\mathrm{Rad}(\delta_1)$ divides $\mathrm{Rad}(\delta_2)$ if and only if $\mathbb{Z}_{\delta_1}$ is a subring of $\mathbb{Z}_{\delta_2}$.

We start with the following lemma (see Remark 2.1 in Noro and Yokoyama (2018)) which tells us about the denominators which can appear in a normal form.

Lemma 3.2. *Let $\delta \in \mathbb{N}_+$, let I be a non-zero ideal in P, let G be its reduced σ-Gröbner basis, and let $f \in P$. Assume that f and G have all coefficients in $\mathbb{Z}_\delta$.*

(a) Every intermediate step of rewriting f via G has all coefficients in $\mathbb{Z}_\delta$.

(b) The polynomial $\mathrm{NF}_\sigma(f)$ has all coefficients in $\mathbb{Z}_\delta$.

Proof. If $f = 0$, the result is trivially true. So we now assume $f \neq 0$. If f can be reduced by G then there exists $\tau \in \mathrm{Supp}(f)$ such that $\tau = t \cdot \mathrm{LT}_\sigma(g)$ for some $g \in G$ and some power-product $t \in \mathbb{T}^n$. Let c be the coefficient of τ in f; by hypothesis $c \in \mathbb{Z}_\delta$. Then the first step of rewriting gives $f_1 = f - c \cdot t \cdot g$ which has all coefficients in $\mathbb{Z}_\delta$. We can now repeat the same argument for rewriting f_1, and so on. The final result, when no further such rewriting is possible, is the normal form of f, and by this same argument it has all coefficients in $\mathbb{Z}_\delta$. Since G is a Gröbner basis the normal form is reached after a finite number of reduction steps, and the result is independent of the choice of reducer at each step. These considerations prove both claims. $\square$

The following example illustrates the lemma.

Example 3.3. Let $P = \mathbb{Q}[x, y]$, let $I = \langle f_1, f_2 \rangle$ where $f_1 = 3x^3 - x^2 + 1$, $f_2 = x^2 - y$, and let $\sigma = \mathtt{DegRevLex}$. The reduced σ-Gröbner basis of I is $G = \{g_1,\, g_2,\, g_3\}$ where $g_1 = y^2 + \frac{1}{3}x - \frac{1}{9}y + \frac{1}{9}$, $g_2 = xy - \frac{1}{3}y + \frac{1}{3}$, and $g_3 = x^2 - y$. We let $f = y^3$ and note that $f, g_1, g_2, g_3 \in \mathbb{Z}_3[x, y]$. We have $\mathrm{NF}_\sigma(f) = -\frac{1}{27}x - \frac{17}{81}y + \frac{8}{81}$, and it is easy to check that the explicit coefficients in the equality

$$f = \mathrm{NF}_\sigma(f) + (xy + \tfrac{1}{9}x + \tfrac{1}{3}y - \tfrac{8}{27})\, g_2 - (y^2 + \tfrac{1}{9}y)\, g_3$$

are the coefficients of a sequence of rewriting steps from f to $\mathrm{NF}_\sigma(f)$. As shown by the lemma, they all lie in $\mathbb{Z}_3$.

This lemma prompts us to make the following definitions.

Definition 3.4. Let $P = \mathbb{Q}[x_1, \ldots, x_n]$.

(a) Given a polynomial $f \in P$, we define the **denominator of** f, denoted by $\mathrm{den}(f)$, to be 1 if $f = 0$, and otherwise the least common multiple of the denominators of the coefficients of f.

(b) Given a non-zero ideal I in P, with reduced σ-Gröbner basis G_σ, we define the **σ-denominator of** I to be $\mathrm{den}_\sigma(I) = \mathrm{lcm}\{\mathrm{den}(g) \mid g \in G_\sigma\}$.

(c) The greatest common divisor of $\mathrm{den}_\sigma(I)$ where σ ranges over all term-orderings is called the **essential denominator** of I.

The following easy example shows that the number δ introduced in Lemma 3.2 depends on σ.

Example 3.5. Let $P = \mathbb{Q}[x, y, z]$, let $I = \langle f \rangle$ where $f = 2x + 3y + 5z$. Depending on the term-ordering chosen, the number δ can be 2, 3 or 5. So we have $\mathrm{den}(f) = 1$, $\mathrm{den}_\sigma(I) = 2$ with $\sigma = \mathtt{DegRevLex}$, and the essential denominator of I is 1.

Now we need one more definition.

Definition 3.6. Let δ be a positive integer, and p be a prime number not dividing δ. We write π_p to denote both the canonical homomorphism $\mathbb{Z}_\delta \longrightarrow \mathbb{F}_p$ and its natural "coefficientwise" extensions to $\mathbb{Z}_\delta[x_1, \ldots, x_n] \longrightarrow \mathbb{F}_p[x_1, \ldots, x_n]$; we call them all **reduction homomorphisms modulo** p.

The following theorem illustrates the importance of the σ-denominator of an ideal.

Theorem 3.7. ***(Reduction modulo*** p ***of Gröbner Bases)***
Let I be a non-zero ideal in $\mathbb{Q}[x_1, \ldots, x_n]$ with reduced σ-Gröbner basis G. Let p be a prime number which does not divide $\mathrm{den}_\sigma(I)$.

(a) The set $\pi_p(G)$ is the reduced σ-Gröbner basis of the ideal $\langle \pi_p(G) \rangle$.

(b) The set of the residue classes of the elements in $\mathbb{T}^n \setminus \mathrm{LT}_\sigma(I)$ is an $\mathbb{F}_p$-basis of the quotient ring $\mathbb{F}_p[x_1, \ldots, x_n]/\langle \pi_p(G) \rangle$.

(c) For every polynomial $f \in \mathbb{Q}[x_1, \ldots, x_n]$ such that $p \nmid \mathrm{den}(f)$ we have the equality $\pi_p(\mathrm{NF}_{\sigma,I}(f)) = \mathrm{NF}_{\sigma,\langle \pi_p(G) \rangle}(\pi_p(f))$.

Proof. We start by proving claim (a). Every polynomial g in G is monic, so $\pi_p(g)$ is monic and $\mathrm{LT}_\sigma(\pi_p(g)) = \mathrm{LT}_\sigma(g)$. Next we show that $\pi_p(G)$ is a reduced σ-Gröbner basis. Let the elements of the Gröbner basis be $G = \{g_1, \ldots, g_s\}$. Let $1 \le i < j \le s$ and let $f_0 = t_j g_i - t_i g_j$ be the S-polynomial of (g_i, g_j). This S-polynomial rewrites to zero via a finite number of steps of rewriting: $f_{k+1} = f_k - c_k \cdot t_k \cdot g_{i_k}$ for $k = 0, 1, \ldots, r-1$. Let $\delta = \mathrm{den}_\sigma(I)$, then f_0 and every g_i have all coefficients in $\mathbb{Z}_\delta$. Lemma 3.2 implies that each c_k is in $\mathbb{Z}_\delta$ and that all coefficients of each f_k are in $\mathbb{Z}_\delta$.

We now show that the S-polynomial of the p-reduced pair $(\pi_p(g_i), \pi_p(g_j))$ rewrites to zero via the set $\pi_p(G)$. First we see that $\pi_p(f_0) = t_j \pi_p(g_i) - t_i \pi_p(g_j)$. Now applying π_p to each rewriting step we get $\pi_p(f_{k+1}) = \pi_p(f_k) - \pi_p(c_k) \cdot t_k \cdot \pi_p(g_{i_k})$. If $\pi_p(c_k) \neq 0$, this is a rewriting step for $\pi_p(f_k)$, otherwise "nothing happens" and we simply have $\pi_p(f_{k+1}) = \pi_p(f_k)$.

This shows that all the S-polynomials of $\pi_p(G)$ rewrite to zero, and hence that $\pi_p(G)$ is a σ-Gröbner basis. Finally we observe that $\mathrm{Supp}(\pi_p(g_i)) \subseteq \mathrm{Supp}(g_i)$ for all $i = 1, \ldots, s$, hence $\pi_p(G)$ is actually the reduced σ-Gröbner basis of the ideal $\langle \pi_p(G) \rangle$.

As already observed, we have $\mathrm{LT}_\sigma(g_i) = \mathrm{LT}_\sigma(\pi_p(g_i))$ for all $i = 1, \ldots, s$, hence claim (b) follows from (a).

For part (c) we let $\delta = \mathrm{lcm}(\mathrm{den}(f), \mathrm{den}_\sigma(I))$. We use the same method as in the proof of part (a) but starting with $f_0 = f$. Once again all rewriting steps have coefficients in $\mathbb{Z}_\delta$, and applying π_p to them we get either a rewriting step for $\pi_p(f)$ or possibly a "nothing happens" step. Therefore the image of the final remainder $\pi_p(\mathrm{NF}(f))$ is the normal form of $\pi_p(f)$. □

The following example illustrates some claims of the theorem.

Example 3.8. We continue the discussion of Example 3.3. We choose $p = 2$ and get $\langle y^2 + x + y + 1,\ xy + y + 1,\ x^2 + y \rangle$ as the (p, σ)-reduction of I. From Theorem 3.7 we know that $\{y^2 + x + y + 1,\ xy + y + 1,\ x^2 + y\}$ is the reduced σ-Gröbner basis of $\langle \pi_p(G) \rangle$.

Theorem 3.7, in particular claim (c), motivates the following definition.

Definition 3.9. In the context of Theorem 3.7, let the reduced σ-Gröbner basis G be $\{g_1, g_2, \ldots, g_s\}$. Then the ideal generated by the set $\pi_p(G) = \{\pi_p(g_1), \ldots, \pi_p(g_s)\}$ in the polynomial ring $\mathbb{F}_p[x_1, \ldots, x_n]$ is called the (p, σ)**-reduction of** I, and will be denoted by $I_{(p,\sigma)}$. Observe that if I is zero-dimensional so is $I_{(p,\sigma)}$.

The following example shows the necessity of considering the *reduced* Gröbner basis in Theorem 3.7.

Example 3.10. Let $P = \mathbb{Q}[x]$, let a be the product of many primes, for instance the product of the first 10^6 prime numbers, and let $I = \langle ax,\, x^2\rangle$. The set $S = \{ax,\, x^2\}$ is a Gröbner basis of I, while the set $G = \{x\}$ is the reduced Gröbner basis of I. Reducing S modulo p where $p \,|\, a$ produces the ideal $\langle x^2\rangle$, while reducing G produces the ideal $\langle x\rangle$.

We conduct a more thorough investigation into (p, σ)-reductions in Abbott et al. (2018b).

3.2. Detection of Suitable Primes

For this entire subsection the ideal I will be zero-dimensional, and since in Theorem 3.7.(b) we have seen that, for a suitable prime p, the set $\mathbb{T}^n \setminus \mathrm{LT}_\sigma(I)$ can be mapped both to a basis of of the ring $\mathbb{Q}[x_1, \ldots, x_n]/I$ and also to a basis of $\mathbb{F}_p[x_1, \ldots, x_n]/I_{(p,\sigma)}$, we are motivated to provide the following definition.

Definition 3.11. Let $P = \mathbb{Q}[x_1, \ldots, x_n]$ with term-ordering σ. Let I be a zero-dimensional ideal in the ring P, with reduced σ-Gröbner basis G; let $\delta = \mathrm{den}_\sigma(I)$. Let the tuple $B = (t_1, t_2, \ldots, t_d) = \mathbb{T}^n \setminus \mathrm{LT}_\sigma(I)$ with elements in increasing σ-order, so necessarily $t_1 = 1$ and $d = \dim_K(P/I)$. We denote the natural image of B in $\mathbb{Q}[x_1, \ldots, x_n]$ by $B_\mathbb{Q}$ and the natural image of B in $\mathbb{F}_p[x_1, \ldots, x_n]$ by B_p. Recall that $B_\mathbb{Q}$ in P/I is a $\mathbb{Q}$-basis of monomials for P/I , and by Theorem 3.7, that B_p is an $\mathbb{F}_p$-basis for $\mathbb{F}_p[x_1, \ldots, x_n]/\langle \pi_p(G)\rangle$ if p is a prime number which does not divide δ.

Let $f \in \mathbb{Z}_\delta[x_1, \ldots, x_n]$. We define the f**-power matrix**, $M_{B_\mathbb{Q}}(f, r)$, to be the $d \times (r+1)$ matrix whose j-th column (for $j = 1, \ldots, r+1$) contains the coordinates of $\mathrm{NF}_{\sigma,I}(f^{j-1})$ in the basis $B_\mathbb{Q}$. Similarly, we define the $\pi_p(f)$**-power matrix** to be $M_{B_p}(\pi_p(f), r)$ the $d \times (r+1)$ matrix whose j-th column contains the coordinates of $\mathrm{NF}_{\sigma,I_{(p,\sigma)}}(\pi_p(f^{j-1}))$ in the basis B_p. We observe that these matrices depend on both σ and the corresponding ideals.

The following proposition contains useful information about reduction of matrices.

Proposition 3.12. *Let $f \in P$ be a polynomial and let $\delta = \mathrm{den}(f) \cdot \mathrm{den}_\sigma(I)$.*

(a) For every r, all the entries of the matrix $M_{B_\mathbb{Q}}(f, r)$ are in $\mathbb{Z}_\delta$.

(b) For every r, we have $\pi_p(M_{B_\mathbb{Q}}(f, r)) = M_{B_p}(\pi_p(f), r)$ for any prime $p \nmid \delta$.

Proof. Claim (a) follows from Lemma 3.2 applied to f^j for $j = 0, 1, \ldots, r$. Claim (b) follows directly from Theorem 3.7.(c). □

3.2.1. Usable primes

We start this subsection with an elementary result which is placed here for the sake of completeness.

Lemma 3.13. *Let $f, g \in \mathbb{Q}[z]$ be monic polynomials such that g divides f, and let $\delta \in \mathbb{N}_+$. If f has coefficients in $\mathbb{Z}_\delta$ then also g has coefficients in $\mathbb{Z}_\delta$.*

Proof. By hypothesis we have a factorization $f = gh$ in $\mathbb{Q}[z]$ for some monic $h \in \mathbb{Q}[z]$. Set $D_f = \operatorname{den}(f)$, $D_g = \operatorname{den}(g)$ and $D_h = \operatorname{den}(h)$; so each of $D_f f$, $D_g g$ and $D_h h$ is a primitive polynomial with integer coefficients. By Gauss's Lemma $(D_g g)(D_h h) = D_g D_h f$ is a primitive polynomial with integer coefficients. Hence $D_f = \pm D_g D_h$; in particular $D_g | D_f$, and consequently $\operatorname{Rad}(D_g) | \operatorname{Rad}(D_f)$. Since $f \in \mathbb{Z}_\delta[z]$ we have $\operatorname{Rad}(D_f) | \operatorname{Rad}(\delta)$, hence also $\operatorname{Rad}(D_g) | \operatorname{Rad}(\delta)$ which implies that $g \in \mathbb{Z}_\delta[z]$. □

We now give a proposition which tells us which primes could appear in the denominator of a minimal polynomial. In the following proposition we use Definition 3.4.(c).

Proposition 3.14. *Let $P = \mathbb{Q}[x_1, \ldots, x_n]$, let I be a zero-dimensional ideal in P, and let f be a polynomial in P.*

(a) for any term-ordering σ let $\delta_\sigma = \operatorname{den}(f) \cdot \operatorname{den}_\sigma(I)$ then the minimal polynomial $\mu_{f,I}(z)$ has all coefficients in $\mathbb{Z}_{\delta_\sigma}$;

(b) let $\delta = \operatorname{den}(f) \cdot D$ where D is the essential denominator of I, then $\mu_{f,I}(z)$ has all coefficients in $\mathbb{Z}_\delta$.

Proof. We prove claim (a). Let $\vartheta_{\bar{f}}$ be the $\mathbb{Q}$-endomorphism of P/I given by multiplication by $\bar{f}$. It is known that $\mu_{f,I}(z) = \mu_{\vartheta_{\bar{f}}}(z)$ (see Remark 4.1.3.(a) in Kreuzer and Robbiano (2016)). Let $\chi_{\vartheta_{\bar{f}}}(z)$ be the characteristic polynomial of the endomorphism $\vartheta_{\bar{f}}$; by definition $\chi_{\vartheta_{\bar{f}}}(z) = \det(z \operatorname{id} - \vartheta_{\bar{f}})$. Next, let $d = \dim_{\mathbb{Q}}(P/I)$, let $B = (1, t_2, \ldots, t_d) = \mathbb{T}^n \setminus \operatorname{LT}_\sigma(I)$, let I_d be the identity matrix of size d, and let $M_B(\vartheta_{\bar{f}})$ be the matrix which represents $\vartheta_{\bar{f}}$ with respect to the basis B. Then we have $\det(z \operatorname{id} - \vartheta_{\bar{f}}) = \det(z\, I_d - M_B(\vartheta_{\bar{f}}))$. The entries of $M_B(\vartheta_{\bar{f}})$ are the coefficients of the representations of $\operatorname{NF}_\sigma(t_i f)$ in the basis B for all $t_i \in B$. They are in $\mathbb{Z}_\delta$ by Lemma 3.2. So we have proved that $\chi_{\vartheta_{\bar{f}}}(z) \in \mathbb{Z}_\delta[z]$. From the Cayley-Hamilton Theorem we deduce that $\mu_{\vartheta_{\bar{f}}}(z)$ is a divisor of $\chi_{\vartheta_{\bar{f}}}(z)$. It follows from Lemma 3.13 that also $\mu_{\vartheta_{\bar{f}}}(z) \in \mathbb{Z}_\delta[z]$.

Claim (b) follows easily from claim (a) and the definition of essential denominator. □

Remark 3.15. To compute the essential denominator one needs know all the possible reduced Gröbner bases of I, in other words one needs to compute the Gröbner Fan of I (see Mora and Robbiano (1993)), but actually computing the fan is practicable only for "fairly simple" ideals.

Example 3.16. Let $P = \mathbb{Q}[x, y]$ and $I = \langle 2x + 3y, y^2 - 4 \rangle$. There are just two possible Gröbner bases: $\{x + \frac{3}{2}y,\ y^2 - 4\}$ and $\{y + \frac{2}{3}x,\ x^2 - 9\}$. Hence the essential denominator is $\gcd(2, 3) = 1$.

Thus we know that the minimal polynomial of any polynomial with integer coefficients has integer coefficients. For instance, let $f = 23x + 17y$ then $\mu_{f,I}(z) = z^2 - 1225$.

The conclusion of the proposition above motivates the following definition.

Definition 3.17. Let $f \in P$ be a polynomial, and let p be a prime number. Then p is called a **usable prime for f with respect to** (I, σ) if it does not divide $\mathrm{den}(f) \cdot \mathrm{den}_\sigma(I)$. If I and σ are clear from the context, we say simply a **usable prime**. It follows from the definition that, for a given input (f, I, σ), there are only finitely many unusable primes, and it is easy to recognize and avoid them.

3.2.2. Good and Bad Primes

In this subsection we refine the definition of usable.

Definition 3.18. Let p be a usable prime for f with respect to (I, σ); consequently, by Proposition 3.14, $\pi_p(\mu_{f,I}(z))$ is well-defined. We say that p is a **good prime for** f if $\mu_{\pi_p(f),I_{(p,\sigma)}}(z) = \pi_p(\mu_{f,I}(z))$, in other words if the minimal polynomial of the p-reduction of f modulo the (p, σ)-reduction of I equals the p-reduction of the minimal polynomial of f modulo I over the rationals. Otherwise, it is called **bad**.

The following simple example illustrates how a prime can be bad even if it is usable.

Example 3.19. Let $P = \mathbb{Q}[x, y]$, let $I = \langle x^2,\, y^2\rangle$, and let $f = x + y$. The set $\{x^2,\, y^2\}$ is a reduced Gröbner basis of I for every term-ordering, $B = (1, x, y, xy)$ is a quotient basis of $\mathbb{Q}[x, y]/I$ regardless of term-ordering. Moreover we have $\mathrm{den}(f) \cdot \mathrm{den}_\sigma(I) = 1$ regardless of term-ordering, and thus every prime number is usable. Over $\mathbb{Q}$ we have $M_B(f, 3) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \end{pmatrix}$. Whence we deduce that $\mu_{f,I}(z) = z^3$. If we change the base field to the finite field $\mathbb{F}_2$, we get $M_B(\pi_2(f), 3) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$ which shows that $\mu_{f,I} = z^2$. It is easy to see that 2 is the only bad prime in this case.

Next we show that there are only finitely many bad primes.

Theorem 3.20. ***(Finitely many bad primes)***
Let $P = \mathbb{Q}[x_1, \ldots, x_n]$, let I be a zero-dimensional ideal in P, let σ be a term-ordering on $\mathbb{T}^n$, let $f \in P$, and p be a usable prime.

(a) Then $\pi_p(\mu_{f,I}(z))$ is a multiple of $\mu_{\pi_p(f),I_{(p,\sigma)}}(z)$.
(b) There are only finitely many bad primes.
(c) The prime p is good if and only if $\deg(\mu_{\pi_p(f),I_{(p,\sigma)}}(z)) = \deg(\mu_{f,I}(z))$.

Proof. To simplify the presentation we let $\mu(z) = \mu_{f,I}(z)$ and $\mu_p(z) = \mu_{\pi_p(f),I_{(p,\sigma)}}(z)$. Let $\mu(z){=}z^r{+}c_{r-1}z^{r-1}{+}\cdots c_0$, and set $\delta = \mathrm{den}(f){\cdot}\mathrm{den}_\sigma(I)$. By Proposition 3.14 we have $\mu(z) \in \mathbb{Z}_\delta[z]$. By the definition of minimal polynomial we have $f^r + c_{r-1}f^{r-1} + \cdots c_0 \in I$. Therefore we have an equality $f^r + c_{r-1}f^{r-1} + \cdots c_0 = \sum h_i g_i$ for certain $h_i \in P$ where $\{g_1, \ldots, g_s\}$ is the reduced σ-Gröbner basis of the ideal I. By Lemma 3.2 we know that each $h_i \in \mathbb{Z}_\delta$. Since p is a usable prime, it follows from Proposition 3.14 that we can apply π_p to get

$$\pi_p(f)^r + \pi_p(c_{r-1})\,\pi_p(f)^{r-1} + \cdots + \pi_p(c_0) = \sum_{i=1}^{s} \pi_p(h_i)\,\pi_p(g_i)$$

which shows that $\pi_p(\mu(f)) \in I_{(p,\sigma)}$, and hence that $\pi_p(\mu(z))$ it is a multiple of $\mu_p(z)$. So claim (a) is proved.

To prove (b) and (c) it suffices to show that only a finite number of usable primes are such that $\pi_p(\mu(z))$ is a non-trivial multiple of $\mu_p(z)$, and we argue as follows. Since r is the degree of $\mu(z)$ we deduce that the matrix $M_{B_{\mathbb{Q}}}(f, r-1)$ has rank r, hence there exists an $r \times r$-submatrix of $M_{B_{\mathbb{Q}}}(f, r-1)$ with non-zero determinant; moreover this determinant lies in $\mathbb{Z}_\delta$, so can be written as $\frac{a}{\delta^s}$ for some non-zero $a \in \mathbb{Z}$ and some $s \in \mathbb{N}$. For any prime p not dividing $a\delta$, the matrix $\pi_p(M_{B_{\mathbb{Q}}}(f, r-1))$ has maximal rank: by Proposition 3.12 we have $\pi_p(M_{B_{\mathbb{Q}}}(f, r-1)) = M_{B_p}(\pi_p(f), r-1)$. Hence for these primes the degree of $\mu_p(z)$ is r, and the conclusion follows. □

The theorem tells us that bad primes are finite in number, and computations confirm that they are very rare (*e.g.* none in the examples described in Subsection 3.4). There appears to be no reasonable guaranteed way to detect bad primes, but the following corollary tells how to easily detect *relatively* bad primes. Using this corollary we can still be misled if there are several bad primes whose modular minimal polynomials all have the same degree (theoretically this is possible, but almost never happens in practice).

Corollary 3.21. ***(Detecting some bad primes)***
In the context of Theorem 3.20, let p_1, p_2 be two usable primes. Let $\mu_1 = \mu_{\pi_{p_1}(f), I_{(p_1,\sigma)}}(z)$ and $\mu_2 = \mu_{\pi_{p_2}(f), I_{(p_2,\sigma)}}(z)$ be the minimal polynomials of the respective modular reductions.

(a) If $\deg(\mu_1) < \deg(\mu_2)$ then p_1 is a bad prime.
(b) If $\deg(\mu_1) = \dim_K(P/I)$ then p_1 is a good prime.

Proof. Claim (a) follows from parts (a) and (c) of Theorem 3.20. Claim (b) follows from Theorem 3.20.(c) since $\dim_K(P/I)$ is an upper bound for the degrees of the minimal polynomials. □

3.3. The Algorithm

Using the results described so far we get the following algorithm. We emphasise a particular aspect of our implementation, the choice of 30-bit primes (on a 64-bit platform): this choice lets us use fast machine integer arithmetic while keeping the number of iterations of the *Main Loop* close to minimal.

Algorithm 3.22. MinPolyQuotModular

```
notation: $P = \mathbb{Q}[x_1, \dots, x_n]$ with term-ordering $\sigma$
Input $I$, a zero-dimensional ideal in $P$, and a polynomial $f \in P$
1 compute the reduced $\sigma$-Gröbner basis of $I$
2 choose a usable prime $p$ — see Definition 3.17.
3 compute $f_p = \pi_p(f)$ and the ideal $I_{(p,\sigma)}$.
4 compute $\mu_p = \mu_{f_p, I_{(p,\sigma)}} \in \mathbb{F}_p[z]$, the minimal polynomial of $f_p$.
5 let $\mu_{\mathrm{crt}} = \mu_p$ and $p_{\mathrm{crt}} = p$.
6 Main Loop:
    6.1 choose a new usable prime $p$.
    6.2 compute the minimal polynomial $\mu_p \in \mathbb{F}_p[z]$.
    6.3 if $\deg(\mu_{\mathrm{crt}}) \neq \deg(\mu_p)$ then
        6.3.1 if $\deg(\mu_{\mathrm{crt}}) < \deg(\mu_p)$ then let $\mu_{\mathrm{crt}} = \mu_p$ and $p_{\mathrm{crt}} = p$.
        6.3.2 continue with next iteration of Main Loop
    6.4 let $\tilde{p}_{\mathrm{crt}} = p \cdot p_{\mathrm{crt}}$, and let $\tilde{\mu}_{\mathrm{crt}}$ be the polynomial whose coefficients are obtained
        by the Chinese Remainder Theorem from the coefficients of $\mu_{\mathrm{crt}}$ and $\mu_p$.
    6.5 compute the polynomial $\mu_{\mathrm{calc}} \in \mathbb{Q}[z]$ whose coefficients are obtained as the
        fault-tolerant rational reconstructions of the coefficients of $\tilde{\mu}_{\mathrm{crt}}$ modulo $\tilde{p}_{\mathrm{crt}}$.
    6.6 were all coefficients "reliably" reconstructed?
        6.6-yes if $\mu_{\mathrm{calc}}(f) \in I$ then return $\mu_{\mathrm{calc}}$
        6.6-no  let $\mu_{\mathrm{crt}} = \tilde{\mu}_{\mathrm{crt}}$ and $p_{\mathrm{crt}} = \tilde{p}_{\mathrm{crt}}$.
    6.7 Continue with next iteration of Main Loop.
Output $\mu_{\mathrm{calc}} \in \mathbb{Q}[z]$, the certified minimal polynomial $\mu_{f,I}$.
```

Proof. The correctness and termination of this algorithm follow from Theorem 3.20, and Corollary 3.21. In particular, note that under our hypothesis, all bad primes give polynomials whose degree is *too low*. This means that if we have $\mu_{\mathrm{calc}}(f) \in I$ (checked in step 6.6-yes) then μ_{calc} is indeed the minimal polynomial, and not a non-trivial multiple. □

Remark 3.23 (Certified answer)**.** When execution enters Step 6.6-yes, the value of μ_{calc} is highly likely to be correct (and will surely be so when p_{crt} is large enough). Since there is nevertheless a small chance of the answer being wrong, either because of a wrong rational reconstruction, or because of a sequence of bad primes with compatible answers, we verify it by explicitly checking that $\mu_{\mathrm{calc}}(f) \in I$.

Some authors, when using modular methods in this area, give algorithms where the answer is correct "with high probability". However, we want to emphasise that our algorithm guarantees that the answer is correct.

Example 3.24 (Example 2.5 continued)**.** Let $P = \mathbb{Q}[x, y]$, $I = \langle x^2 - \frac{1}{7}y^2 - 5,\ y^2 + 4x - \frac{7}{2} \rangle$, and $f = 3x - 2y$. Two computations modulo $p_1 = 1073741831$, and $p_2 = 1073741833$ give $\mu_{p_1} = z^4 - 460175067z^3 + 525914233z^2 - 306782542z - 191739221$ and $\mu_{p_2} = z^4 - 153391687z^3 - 131478725z^2 - 460174233z - 421826757$. Notice that they have the same degree. Their CRT combination, modulo $\tilde{p}_{\mathrm{crt}} = p_1 p_2$, is $\tilde{\mu}_{\mathrm{crt}} =$
$z^4 - 164703074540959457z^3 + 352935159730627282z^2 - 494109223622877543z + 123527305905719987$
(in CoCoA this is computed by `CRTPoly(mp1,p1, mp2,p2)`). Its rational reconstruction is $\mu_{\mathrm{calc}} = z^4 + \frac{24}{7}z^3 - \frac{6527}{49}z^2 + \frac{5868}{7}z + \frac{10967}{28}$ (in CoCoA `RatReconstructPoly(`$\mu_{\mathrm{crt}}, p_{\mathrm{crt}}$`)`). Then we verify that $\mu_{\mathrm{calc}}(f) \in I$, and we may conclude that μ_{calc} is indeed $\mu_{f,I}$. Note that the rational reconstruction algorithm requires a modulus a bit larger than might seem necessary so that the reconstructed values are "reliable" (*i.e.* likely correct).

Remark 3.25 (Verification)**.** Termination of the *Main Loop* in Algorithm 3.22 depends on the test $\mu_{\mathrm{calc}}(g) \in I$ in step MinPolyQuotModular-6.6-yes; however evaluating $\mu_{\mathrm{calc}}(g)$ modulo I is typically computationally expensive compared to the cost of a single iteration. For this reason, in step MinPolyQuotModular-6.5 we use the fault-tolerant rational reconstruction implemented in CoCoA (see Abbott (2017)) which gives also an indication whether the reconstructed rational is "reliable" (*i.e.* heuristically probably correct). This is a computationally cheap criterion which surely indicates "reliable" almost as soon as $\tilde{p}_{\mathrm{crt}}$ becomes large enough to allow correct reconstruction, while also almost certainly indicating "not reliable" before then.

Once a good prime has been picked, μ_{crt} will have the correct degree, and thereafter the degree check in step MinPolyQuotModular-6.3 ensures that only results from good primes are used; in this situation our fault-tolerant reconstruction is equivalent to Monagan's MQRR (Monagan, 2004).

Even though it has happened only extremely rarely, we have encountered "reliable" reconstructions which did not return the exact answer, so by default CoCoA truly verifies that $\mu_{\mathrm{calc}}(g)$ is in I. Even if performed just once this operation may be quite costly, so we also offer a partial verification: calling `MinPoly(f,I,z,N)` verifies for N random primes p that $\pi_p(\mu_{\mathrm{calc}}(\pi_p(g))$ is in $I_{(p,\sigma)}$ (in the timings table below, we check for 3 primes).

Remark 3.26 (No Gröbner basis)**.** A disadvantage of Algorithm 3.22 is that it needs a Gröbner basis over $\mathbb{Q}$, requiring a potentially costly computation. We can make a faster heuristic variant of the algorithm by working directly with the given generators for I. Let G' be the set of given generators. We shall skip all unusable primes which divide $\mathrm{den}(G')$.

In steps MinPolyQuotModular-3 and 4 we use the ideal $\langle \pi_p(G') \rangle$ instead of $I_{(p,\sigma)}$. In the *Main Loop* we skip step MinPolyQuotModular-6.3, since there are no guarantees on the degrees of bad μ_p. For instance, in Example 3.10, the minimal polynomial of x modulo $I = \langle ax, x^2 \rangle$ modulo all "small" primes, is z^2, instead of z.

Thus, we keep all the μ_p but when using the chinese remainder theorem to combine, we take only those polynomials having the same degree as the current μ_p.

In step MinPolyQuotModular-6.6-yes we return directly μ_{calc} skipping the check that $\mu_{\mathrm{calc}}(g)$ is in I, since we cannot verify the answer because we want to avoid computing Gröbner basis.

There are only finitely many primes giving a bad μ_p. We can see this by picking some term-ordering σ, and tracing through the steps to compute the reduced σ-Gröbner basis from the generators G'. Any prime which divides a denominator or a leading coefficient at any point in the computation may give a bad μ_p; to these we add the (finitely many) bad primes for that reduced Gröbner basis. All remaining primes will give a good μ_p.

In conclusion, what do we do in CoCoA? We recall that CoCoA, whenever computing the Gröbner basis G of an ideal I, stores G within the representation of I. This means that if G has already been computed (`HasGBasis(I)` gives true), then we can use it, and the answer of `MinPoly(f,I,z)`, is fully guaranteed. This happens in all the functions described in Section 4.

In case of a direct call with no precomputed Gröbner basis the implementation of `MinPoly(f,I,z)` follows Remark 3.26 (giving a warning, if the user sets high verbosity with `SetVerbosityLevel(80)`). A partial verification is performed by `MinPoly(f,I,z,N)` which verifies over N more primes.

In practice, we have observed no significant advantage in skipping the computation of the Gröbner basis, but we keep this code for flexibility and further investigations.

3.4. Timings: Computing Minimal Polynomials over $\mathbb{Q}$

In this subsection we present some timings for the computation of minimal polynomials of elements in zero-dimensional affine $\mathbb{Q}$-algebras.

The column **Example** gives the reference number to the examples listed below. The column **GB** gives the times to compute the `DegRevLex`-Gröbner basis. Under the heading **MinPoly** the column $\mathbb{Q}$ gives the times of the direct computation over $\mathbb{Q}$ using

Algorithm 2.10) MinPolyQuotDef; under the sub-heading **Modular** the first sub-column gives the times of the computation using Algorithm 3.22 MinPolyQuotModular (which internally uses MinPolyQuotDef for each modular computation) with full verification over $\mathbb{Q}$, *i.e.* checking that the reconstructed polynomial actually vanishes on the input polynomial, while the second sub-column gives the time with a heuristic verification over $\mathbb{F}_p$ for 3 random primes between 10^9 and $2 \cdot 10^9$; the third sub-column ($\#p$) gives the number of primes used for the reconstruction. In the tables, all times are in seconds.

The columns **coeff** and **deg** give an indication of the size of the minimal polynomial: the first expresses the maximum magnitude of the numerators and denominators of the coefficients, and the second is the degree.

Remark 3.27. CoCoA also offers `RingTwinFloat` arithmetic, an implementation of *heuristically guaranteed* floating point numbers (Abbott, 2012). We have also tried our algorithms using this representation of $\mathbb{Q}$, but the modular approach gave us better timings.

Table 2. Timings over the rationals

Example	**GB**	**dim$_K$**	**MinPoly**						
			deg	$\mathbb{Q}$	**Modular**				**coeff**
					$\mathbb{Q}$-*verif*		*3-verif*		
					tot	*verif*	*verif*		
	time			*time*	*time*	*time*	*time*	$\#p$	
3.28 f	0.15	117	116	> 600	15.25	4.55	0.26	64	$10^{389}, 10^{188}$
3.29 x	0.00	108	107	47.86	0.39	0.05	0.04	12	$10^{93}, 10^{0}$
f			108	224.06	1.31	0.11	0.06	25	$10^{210}, 10^{0}$
3.30 f	0.00	144	144	> 600	3.77	0.21	0.17	38	$10^{330}, 10^{0}$
3.31 f	0.00	120	120	45.43	0.67	0.23	0.13	9	$10^{64}, 10^{0}$
3.32 f	0.00	720	720	> 600	172.54	14.51	7.80	58	$10^{503}, 10^{0}$
3.33 z	0.00	230	230	233.24	0.39	0.10	0.09	5	$10^{29}, 10^{4}$
3.34 z	0.42	149	149	89.32	11.30	10.42	0.30	7	$10^{33}, 10^{19}$
f			149	> 600	18.12	12.70	0.38	30	$10^{234}, 10^{19}$
3.35 f	0.33	55	55	5.33	0.67	0.24	0.03	15	$10^{108}, 10^{12}$
3.36 y	0.00	378	252	510.85	3.45	1.39	1.44	3	$10^{11}, 10^{0}$
f			252	> 600	20.43	2.37	1.66	26	$10^{222}, 10^{0}$

Example 3.28. This is an example with no particular structure.

Let $P = \mathbb{Q}[x, y, z, t]$, let $g_1 = xyzt + 83x^3 + 73y^2 - 85z^2 - 437t$, $g_2 = x^3zt + z - t$, $g_3 = zt^2 + 76x + 94y^2 - 324z^3 - 255t^4$, $g_4 = y^2z + 625x + 26t^3$, and let $I = \langle g_1,\, g_2,\, g_3,\, g_4\rangle$. Let $f = t^2 + 5z$.

The following two examples use ideals which are complete intersections; their reduced Gröbner bases are straightforward to compute.

Example 3.29. Let $P = \mathbb{Q}[x, y, z, t]$, let $g_1 = x^4 + 83x^3 + 73y^2 - 85z^2 - 437t,\ g_2 = y^3 - x,\ g_3 = z^3 + z - t,\ g_4 = t^3 - 324z^2 + 94y^2 + 76x$. Let $I = \langle g_1,\ g_2,\ g_3,\ g_4\rangle$ and $f = 2x + 3y - 4z + 12t$.

Example 3.30. Let $P = \mathbb{Q}[x, y, z, t]$, let $g_1 = x^4 + 83z^3 + 73y^2 - t^2 - 437t,\ g_2 = y^3 - z - t,\ g_3 = z^3 + x - t,\ g_4 = t^4 - 12z^2 + 77y^2 + 15x$. Let $I = \langle g_1, g_2, g_3, g_4\rangle$ and $f = x - 3y - 12z + 62t$.

Example 3.31. This is an example which uses the defining ideal of the splitting algebra of a polynomial of degree 5.

We let $P = \mathbb{Q}[a_1, a_2, a_3, a_4, a_5]$, and for $j = 1, \ldots, 5$ let s_j be the elementary symmetric polynomial in the indeterminates a_1, a_2, a_3, a_4, a_5. Then we introduce the ideal $I = \langle s_1,\ s_2,\ s_3,\ s_4 + 1,\ s_5 - 2\rangle$ which is the defining ideal of the splitting algebra of the polynomial $x^5 - x - 2$. We let $f = a_1 + 2a_2 + 3a_3 + 4a_4 + 5a_5$.

Example 3.32. This is an example which uses the defining ideal of the splitting algebra of a polynomial of degree 6 (see also 2.17).

We let $P = \mathbb{Q}[a_1, a_2, a_3, a_4, a_5, a_6]$, and for $j = 1, \ldots, 6$ let s_j be the elementary symmetric polynomial in the indeterminates $a_1, a_2, a_3, a_4, a_5, a_6$. The ideal $I = \langle s_1,\ s_2,\ s_3,\ s_4,\ s_5 - 7,\ s_6 - 1\rangle$ is the defining ideal of the splitting algebra of the polynomial $x^6 - 7x + 1$. We let $f = a_1 + 2a_2 + 3a_3 + 4a_4 + 5a_5 + 6a_6$.

Example 3.33. In this example we let $g_1 = z^7 - 3x - y,\ g_2 = y^5z - 5057x^2 - 2,\ g_3 = x^6y - x - z + 14$, and $I = \langle g_1,\ g_2,\ g_3\rangle$.

Example 3.34. In this example we let $f_1 = x^5 - y - 3z,\ f_2 = xy^5 - 5057z^2 - 2,\ f_3 = yz^5 - x - z + 14$ and $J_1 = \langle f_1,\ f_2,\ f_3\rangle$. Then we let $g_1 = x^2 - y - 3z, g_2 = xy - 5z^2 - 12, g_3 = z^3 - x - y + 4$ and $J_2 = \langle g_1, g_2, g_3\rangle$. Then we let $I = J_1 \cap J_2$. We do not give explicit generators of I since they are cumbersome. Finally we let $f = 7x - 5y + 2z$.

Example 3.35. This is a simplified version of Example 3.34, in the sense that J_2 and f are the same. Instead we let $f_1' = x^3 - y - 3z, f_2' = xy^3 - 5057z^2 - 2, f_3' = yz^4 - x - z + 14$ and let $J_1 = \langle f_1',\ f_2',\ f_3'\rangle$.

Example 3.36. This is an example of a non-reduced $\mathbb{Q}$-algebra. Let $P = \mathbb{Q}[x, y, z]$, and $g_1 = (z^7 - z - 1)^2$, $g_2 = (x^2 - yz)^3$, $g_3 = x^9 - x - 1$; let $I = \langle g_1,\ g_2,\ g_3\rangle$ and linear form $f = 2x - 5y + 7z$.

Remark 3.37. From Table 2 it is clear that the least common denominators of the coefficients of the minimal polynomials are often quite small. This is a natural consequence of Prop. 3.14.(b).

4. Uses of Minimal Polynomials

In this section we let K be a field of characteristic zero or a perfect field of characteristic $p > 0$ having effective p-th roots, and let I be a zero-dimensional ideal in the polynomial ring $P = K[x_1, \ldots, x_n]$. We start the section by recalling a useful definition.

Definition 4.1. Let $f \in P$ be a non-zero polynomial with positive degree. We define the **square-free part**, $\mathrm{sqfree}(f)$, to be the product of all distinct irreducible factors of f (which are defined up to a constant factor). Equivalently, $\mathrm{sqfree}(f)$ is a generator of the radical of the principal ideal generated by f. If f is univariate, and the coefficient field has characteristic zero then $\mathrm{sqfree}(g)$ is $\frac{g}{\gcd(g,g')}$ up to a constant factor; if the characteristic is positive then we can use the algorithm described in Proposition 3.7.12 (Kreuzer and Robbiano, 2008).

In the next proposition we collect important results which will be used throughout the entire section.

Proposition 4.2. *Let K be a perfect field, let $P = K[x_1, \ldots, x_n]$, let I be a zero-dimensional ideal in P, and let $R = P/I$.*

(a) If K is infinite:

(a_1) If I is radical, $\deg(\mu_{\ell,I}(z)) = \dim_K(P/I)$ for the generic linear form $\ell \in P$.

(a_2) If I is maximal, $\bar{\ell}$ is a primitive element of the field P/I where $\ell \in P$ is the generic linear form.

(b) If K is finite: if I is maximal, there exists $f \in P$ such that $\bar{f}$ is a primitive element of the field P/I.

Proof. To prove claim (a) we observe that (a_2) is a special case of (a_1), hence we prove (a_1). Since I is radical and K is infinite, the Shape Lemma (*e.g.* see Theorem 3.7.25 in Kreuzer and Robbiano (2008)) guarantees the existence of a linear change of coordinates which brings I into normal x_n-position, hence after such a transformation the last indeterminate has squarefree minimal polynomial of degree $\dim_K(P/I)$. Equivalently, the generic linear change of coordinates yields a situation where the minimal polynomial of the last indeterminate is squarefree and has degree $\dim_K(P/I)$. If I is maximal then this polynomial is necessarily irreducible. This is exactly what Algorithm 4.16 tries to achieve via randomization in step IsMaximal-5 (the *Second Loop*).

The proof of claim (b) follows from the well-known fact that the multiplicative group of a finite field $L = P/I$ is cyclic, so that if a is a generator of $L\backslash\{0\}$ we have $L = K[a]$ which implies that $\deg(\mu_{a,I}(z)) = \dim_K(P/I)$. We then choose $f \in P$ such that $\bar{f} = a$. □

The following example shows that if K is finite, and I is radical but not maximal then it is possible that no element $f \in P$ exists such that $\deg(\mu_{\ell,I}(z)) = \dim_K(P/I)$.

Example 4.3. Let $K = \mathbb{F}_2$, let $P = K[x, y]$, $I = \langle x^2 + x,\ y^2 + y\rangle$. Then we can write I as an intersection of maximal ideals: $I = M_1 \cap M_2 \cap M_3 \cap M_4$ where $M_1 = \langle x, y\rangle$, $M_2 = \langle x, y + 1\rangle$, $M_3 = \langle x + 1, y\rangle$, $M_4 = \langle x + 1, y + 1\rangle$. So whatever element $f \in P$ we choose, we have $\deg(\mu_{f,I}(z)) \leq 2$ while $\dim_K(P/I) = 4$.

4.1. IsRadical and Radical for a Zero-Dimensional Ideal

The goal of this subsection is to describe algorithms for checking if I is radical, and for computing the radical of I. We need the following results.

Proposition 4.4. *Let K be a perfect field, let $P = K[x_1, \ldots, x_n]$, let I be a zero-dimensional ideal in P, let $f_i(x_i)$ be such that $I \cap K[x_i] = \langle f_i(x_i)\rangle$ for $i = 1, \ldots, n$, and let $g_i = \mathrm{sqfree}(f_i(x_i))$. Then we have the equality $\sqrt{I} = I + \langle g_1, \ldots, g_n\rangle$.*

Proof. By Proposition 3.7.1 in Kreuzer and Robbiano (2008), the polynomials $f_i(x_i)$ are non-zero. Since the ideal $J = I + \langle g_1, \ldots, g_n \rangle$ satisfies $I \subseteq J \subseteq \sqrt{I}$, we have $\sqrt{J} = \sqrt{I}$. By Proposition 3.7.9 in Kreuzer and Robbiano (2008) we have $\gcd(g_i, g_i') = 1$ for all $i = 1, \ldots, n$, hence the conclusion follows from Seidenberg's Lemma (see Proposition 3.7.15 in Kreuzer and Robbiano (2008)). □

Since $I \cap K[x_i] = \langle \mu_{x_i,I}(x_i) \rangle$, the above proposition can be rewritten as follows.

Corollary 4.5. *Let K be a perfect field, let $P = K[x_1, \ldots, x_n]$, let I be a zero-dimensional ideal in P, and let $g_i = \operatorname{sqfree}(\mu_{x_i,I}(x_i))$. Then we have $\sqrt{I} = I + \langle g_1, \ldots, g_n \rangle$.*

The following proposition shows that in some cases it is particularly easy to show that an ideal is radical.

Proposition 4.6. *Let K be a perfect field, let I be a zero-dimensional ideal in the polynomial ring $P = K[x_1, \ldots, x_n]$, and let $f \in P$. If the polynomial $\mu_{f,I}(z)$ is squarefree and $\deg(\mu_{f,I}(z)) = \dim_K(P/I)$ then I is a radical ideal.*

Proof. Consider the K-algebra homomorphism $\alpha_f \colon K[z]/\langle \mu_{f,I}(z) \rangle \to P/I$ sending $\bar{z} \mapsto \bar{f}$ which is injective by definition. Since $\dim_K(K[z]/\langle \mu_{f,I}(z) \rangle) = \deg(\mu_{f,I}(z)) = \dim_K(P/I)$, then α_f is also surjective and hence an isomorphism. By assumption, the polynomial $\mu_{f,I}(z)$ is squarefree and hence $P/I \cong K[z]/\langle \mu_{f,I}(z) \rangle$ is a reduced K-algebra which means that I is a radical ideal. □

The following algorithm determines whether a zero-dimensional ideal is radical.

```
Algorithm 4.7. IsRadical0Dim
notation: $K$ a perfect field and $P = K[x_1, \ldots, x_n]$
Input $I$, a zero-dimensional ideal in $P$
1 compute $d = \dim_K(P/I)$
2 Main Loop: for $i = 1, \ldots, n$ do
    2.1 compute $\mu = \mu_{x_i,I}$
    2.2 if $\mu$ is not square-free then return false
    2.3 if $\deg(\mu) = d$ then return true
3 return true
Output true/false indicating whether $I$ is radical or not.
```

Proof. Clearly, the algorithm ends after a finite number of steps and its correctness follows from Corollary 4.5 and Proposition 4.6 □

Similarly, we have an algorithm for computing the radical of a zero-dimensional ideal.

```
Algorithm 4.8. Radical0Dim
notation: $K$ a perfect field and $P = K[x_1, \dots, x_n]$
Input  $I$, a zero-dimensional ideal in $P$
1  let $J = I$ and compute $d = \dim_K(P/J)$
2  Main Loop: for $i = 1, \dots, n$ do
    2.1  compute $\mu = \mu_{x_i,J}$
    2.2  if $\mu$ is not square-free then
        2.2.1  let $\mu = \mathrm{sqfree}(\mu)$
        2.2.2  let $J = J + \langle \mu(x_i) \rangle$
        2.2.3  compute $d = \dim_K(P/J)$
    2.3  if $\deg(\mu) = d$ then return $J$
3  return $J$
Output  $J$: the radical of $I$
```

Proof. Clearly, the algorithm ends after a finite number of steps and its correctness follows from Proposition 4.6 and Corollary 4.5. □

Example 4.9. One might hope for a fast, randomized heuristic version of this algorithm: instead of the *Main Loop* we pick a random linear form ℓ, and set $\mu = \mathrm{sqfree}(\mu_{\ell,J})$, and then update $J = J + \langle \mu(\ell) \rangle$. The example here shows that a single random linear form is not always sufficient; indeed, for this example n linearly independent linear forms must be used before the correct result is obtained.

Let K be a field, let $P = K[x_1, \dots, x_n]$ with $n \geq 2$, and let the ideal $I = \langle x_1, \dots, x_n \rangle^2$. Now let $\ell \in P$ be any non-zero linear form. Clearly $\mu_{\ell,I}(z) = z^2$. Hence $2 = \deg(\mu_{\ell,I}(z)) < \dim(P/I) = n + 1$, and adding $\langle \ell \rangle$ to I does not yield $\langle x_1, \dots, x_n \rangle$.

Remark 4.10 (Timeout on Gröbner basis)**.** In step Radical0Dim-2.2.3 we update the value of d. In practice this step can be very costly. Since the purpose of d is to let the algorithm finish before having completed all iterations of the *Main Loop*, and the since update in step 2.2.3 must reduce the value of d, we can safely skip the update if the computation of $\dim_K(P/J)$ takes too long: in our implementation in CoCoALib we have set a heuristic time limit for this step. In the worst case the algorithm simply performs all iterations, even though theoretically it may have been able to stop at an earlier iteration. The time limit we chose is equal to half the expected time for all the remaining minimal polynomials, based on the average time for the first ones.

Example 4.11. Let $I = \langle x^3 + 2x^2y - 2yz^2, 5y^4 - 4y^3z + 3yz^3, 5z^4 + 3xy^2 - 8yz^2 \rangle \in \mathbb{Q}[x, y, z]$. We compute the radical of I quite quickly, ≈ 0.1s, by adding $\mathrm{sqfree}(\mu_{x,I}(x))$, $\mathrm{sqfree}(\mu_{y,I}(y))$, $\mathrm{sqfree}(\mu_{z,I}(z))$. However, its Gröbner basis is much harder to compute (≈ 10s):

$$\{\ x^3 - \tfrac{6228}{3125}x^2 - \tfrac{6488}{375}xy - \tfrac{17648}{375}y^2 + \tfrac{5502}{625}xz + \tfrac{67124}{1875}yz - \tfrac{1334032}{28125}z^2 - \tfrac{17208}{15625}x + \tfrac{11169272}{140625}y - \tfrac{120856}{3125}z,$$

$$x^2y + \tfrac{6903}{6250}x^2 + \tfrac{3469}{375}xy + \tfrac{8224}{375}y^2 - \tfrac{2976}{625}xz - \tfrac{66881}{3750}yz + \tfrac{681758}{28125}z^2 + \tfrac{2079}{31250}x - \tfrac{5702572}{140625}y + \tfrac{12037}{625}z,$$

$$xy^2 - \tfrac{6903}{12500}x^2 - \tfrac{623}{250}xy - \tfrac{2368}{375}y^2 + \tfrac{1488}{625}xz + \tfrac{7637}{1500}yz - \tfrac{36997}{9375}z^2 + \tfrac{75909}{62500}x + \tfrac{294194}{46875}y - \tfrac{36889}{6250}z,$$

$$y^3 - \tfrac{297}{25000}x^2 - \tfrac{177}{500}xy - \tfrac{128}{125}y^2 - \tfrac{144}{625}xz + \tfrac{119109}{125000}yz + \tfrac{110991}{156250}z^2 + \tfrac{166419}{3125000}x + \tfrac{36036}{390625}y + \tfrac{2673}{312500}z,$$

$$x^2z + \tfrac{1116}{625}x^2 + \tfrac{212}{25}xy - \tfrac{8696}{675}y^2 + \tfrac{1168}{375}xz + \tfrac{41912}{3375}yz + \tfrac{39034}{1875}z^2 - \tfrac{1512}{3125}x - \tfrac{339056}{9375}y - \tfrac{17348}{1125}z,$$

$xyz - \frac{558}{625}x^2 - \frac{106}{25}xy - \frac{64}{25}y^2 + \frac{72}{125}xz + \frac{44}{25}yz - \frac{14368}{1875}z^2 + \frac{5124}{3125}x + \frac{117536}{9375}y - \frac{1158}{625}z,$
$y^2z + \frac{54}{625}x^2 + \frac{3}{25}xy - \frac{32}{25}y^2 - \frac{36}{125}xz + \frac{162}{3125}yz + \frac{21552}{15625}z^2 - \frac{30258}{78125}x - \frac{52416}{78125}y - \frac{972}{15625}z,$
$xz^2 - \frac{27}{125}x^2 - \frac{22}{5}xy - \frac{16}{5}y^2 + \frac{18}{25}xz + \frac{247}{100}yz - \frac{2773}{375}z^2 + \frac{81}{2500}x + \frac{22832}{1875}y - \frac{66}{25}z,$
$yz^2 + \frac{27}{250}x^2 + \frac{3}{5}xy - \frac{8}{5}y^2 - \frac{9}{25}xz + \frac{81}{1250}yz + \frac{1638}{3125}z^2 - \frac{15129}{31250}x - \frac{13104}{15625}y - \frac{243}{3125}z,$
$z^3 + \frac{27}{200}x^2 + \frac{3}{4}xy - \frac{1519}{1000}yz + \frac{819}{1250}z^2 - \frac{15129}{25000}x - \frac{3276}{3125}y - \frac{243}{2500}z \}$

This shows the advantage of using the CoCoALib timeout mechanism to interrupt step Radical0Dim-2.2.3 when it takes too long.

4.2. IsMaximal, and IsPrimary for a Zero-Dimensional Ideal

In this subsection, we describe methods for checking if a zero-dimensional ideal is primary or is maximal. To do this we use different strategies depending on the characteristic of the base field. In particular, when K is a finite field with q elements we can use a specific tool, namely a K-vector subspace of $R = P/I$, called the **Frobenius space of** R. The main property is that its dimension is exactly the number of primary components of I. For the definition and basic properties of Frobenius spaces we refer to Section 5.2 in Kreuzer and Robbiano (2016). For convenience, we recall the definition here.

Definition 4.12. Let K be a finite field with characterstic p, and let $q = p^e$ be a power of p. Let $R = P/I$ be a zero-dimensional K-algebra.

(a) The map $\Phi_q : R \to R$ defined by $a \mapsto a^q$ is a K-linear endomorphism of R called the q-**Frobenius endomorphism** of R.

(b) The fixed-point space of R with respect to Φ_q, namely the set $\{f \in R \mid f^q - f = 0\}$, is called the q-**Frobenius space** of R, and is denoted by $\mathrm{Frob}_q(R)$.

Remark 4.13. Note that here we define the generalized Frobenius endomorphism $a \mapsto a^q$ instead of the classical Frobenius endomorphism, $a \mapsto a^p$. The generalized endomorphism is just the classical endomorphism iterated. In this article we shall always take $q = \#K$.

The following proposition describes some features of minimal polynomials when the zero-dimensional ideal I is primary or maximal.

Proposition 4.14. *Let I be a zero-dimensional ideal in $P = K[x_1, \ldots, x_n]$.*

(a) If I is primary then for any $f \in P$ its minimal polynomial $\mu_{f,I}(z)$ is a power of an irreducible polynomial.

(b) If I is maximal then for any $f \in P$ its minimal polynomial $\mu_{f,I}(z)$ is irreducible.

Proof. We use the same argument as in the proof of Proposition 4.6, so that we get an in injective K-algebra homomorphism $K[z]/\langle\mu_{f,I}(z)\rangle \to P/I$.

Now we prove claim (a). If I is primary then the only zero-divisors of P/I are nilpotent, hence the same property is shared by $K[z]/\langle\mu_{f,I}(z)\rangle$ which implies that $\mu_{f,I}(z)$ is a power of an irreducible element.

Analogously, if I is maximal then P/I is a field, hence $K[z]/\langle\mu_{f,I}(z)\rangle$ is an integral domain which concludes the proof. □

We have a sort of converse of the above proposition.

Proposition 4.15. *Let I be a zero-dimensional ideal in $P = K[x_1, \ldots, x_n]$, and let $f \in P$ be such that* $\deg(\mu_{f,I}(z)) = \dim_K(P/I)$.

(a) If $\mu_{f,I}(z)$ is a power of an irreducible factor then I is a primary ideal.

(b) If $\mu_{f,I}(z)$ is irreducible then I is a maximal ideal.

Proof. As in the proof of Proposition 4.14 we have an injective K-algebra homomorphism $K[z]/\langle \mu_{f,I}(z)\rangle \to P/I$. The assumption that $\deg(\mu_{f,I}(z)) = \dim_K(P/I)$ implies that this endomorphism is actually an isomorphism. Now if $K[z]/\langle \mu_{f,I}(z)\rangle$ has only one maximal ideal, the same property is shared by P/I which implies that I is a primary ideal and claim (a) is proved. Analogously, if $K[z]/\langle \mu_{f,I}(z)\rangle$ is a field, then also P/I is a field which means that I is a maximal ideal. □

4.3. IsMaximal

Our next goal is to check whether an ideal I in P is maximal, and the following algorithm provides an answer. Note that is a true algorithm when K is finite, whereas the termination is only heuristically guaranteed when K is infinite.

```
Algorithm 4.16. IsMaximal
notation: $K$ a perfect field and $P = K[x_1,\dots,x_n]$
Input  $I$, an ideal in $P$
1 if $I$ is not zero-dimensional, return false
2 compute $d = \dim_K(P/I)$
3 First Loop: for $i = 1,\dots,n$ do
    3.1 compute $\mu = \mu_{x_i,I}$
    3.2 if $\mu$ is reducible then return false
    3.3 if $\deg(\mu) = d$ then return true
4 if $K$ is finite then
    4.1 compute $s = \dim_K(\mathrm{Frob}_q(P/I))$
    4.2 if $s = 1$ return true else return false
5 (else $K$ is infinite) Second Loop: repeat
    5.1 pick a random linear form $\ell \in P$
    5.1 compute $\mu = \mu_{\ell,I}$
    5.2 if $\mu$ is reducible then return false
    5.3 if $\deg(\mu) = d$ then return true
Output  true/false indicating the maximality of $I$.
```

Proof. Let us show the correctness. In step 3.2, if μ is reducible, we conclude from Proposition 4.14.(b). In step 3.3, since μ is irreducible, if $\deg(\mu) = d$ then we conclude from Proposition 4.15.(b). If the *First Loop* completes without returning an answer, all polynomials $\mu_{x_i,I}(x_i)$ are irreducible and belong to I, hence I is radical by Seidenberg's Lemma (see Kreuzer and Robbiano (2008), Proposition 3.7.15 and Corollary 3.7.16). Now we know that I is radical, we examine the two cases below.

First we consider the case when K is finite. Then the ideal I is maximal if and only if $\dim_K(\mathrm{Frob}_q(P/I)) = 1$ (see Kreuzer and Robbiano (2016), Theorem 5.2.4.(b)). Therefore, when K is finite, steps 4.1 and 4.2 show that the algorithm is correct and terminates.

Now we consider the case when K is infinite. In step 5.2 if the minimal polynomial μ is reducible, Proposition 4.14.(b) tells us that I is not maximal. In step 5.3 we know that the polynomial μ is irreducible, so if $\deg(\mu) = d$, Proposition 4.15.(b) tells us that I is maximal. We conclude that also in this case the algorithm is correct. Its termination follows heuristically from Proposition 4.2.(a). $\square$

Can we make this into a proper deterministic algorithm when K is infinite? The following remark answers this question.

Remark 4.17. If K is infinite, we can substitute the *Second Loop* with a check that in the special family of linear forms described in Lemma 2.1 in Rouillier (1999) there is one whose minimal polynomial has degree d. In this way the algorithm becomes deterministic, however the coefficients of the linear forms tend to become large and the computation more expensive.

Remark 4.18. Since the computation of the Frobenius space in step IsMaximal-4.1 might be costly, one could be tempted to first try a few random linear forms (as in the *Second Loop*). However, our experiments show that computing the minimal polynomial for a random linear form has a computational cost very similar to that for the Frobenius space, while potentially furnishing less information. In summary, there is no benefit from inserting such a "heuristic step" just before IsMaximal-4.1.

4.4. IsPrimary for a Zero-Dimensional Ideal

The goal of this subsection is to check whether a zero-dimensional ideal I in P is primary. The structure of the following algorithm is very similar to the structure of Algorithm 4.16. In particular, it is important to observe that also in this case it is a true algorithm when K is finite, whereas the termination is only heuristically guaranteed when K is infinite.

Algorithm 4.19. IsPrimary0Dim

```
notation: $P = K[x_1, \ldots, x_n]$
Input $I$, a zero-dimensional ideal in $P$
1 let $J = I$ and compute $d = \dim_K(P/J)$
2 First Loop: for $i = 1$ to $n$ do
    2.1 compute $\mu = \mu_{x_i,J}$
    2.2 factorize $\mu$
    2.3 if $\mu$ is not a power of an irreducible factor return false
    2.4 if $\deg(\mu) = d$ then return true
    2.5 if $\mu$ is not square-free then
        2.5.1 let $\mu = \mathrm{sqfree}(\mu)$
        2.5.2 let $J = J + \langle \mu(x_i) \rangle$
        2.5.3 compute $d = \dim_K(P/J)$
        2.5.4 if $\deg(\mu) = d$ then return true
3 if $K$ is finite then
    3.1 compute $s = \dim_K(\mathrm{Frob}_q(P/I))$
    3.2 if $s = 1$ return true else return false
4 (else $K$ is infinite) Second Loop: repeat
    4.1 pick a random linear form $\ell \in P$
    4.2 compute $\mu = \mu_{\ell,J}$
    4.3 if $\mu$ is reducible return false
    4.4 if $\deg(\mu) = d$ then return true
Output true/false indicating whether $I$ is primary or not.
```

Proof. Let us show the correctness. In the *First Loop* we work with an ideal J such that $\sqrt{J} = \sqrt{I}$, because of the change we might perform in step 2.5. In particular J is primary if and only if I is primary. Moreover, at the end of the *First Loop*, $J = \sqrt{I}$ by Seidenberg's Lemma (see the proof of Algorithm 4.16).

In step 2.3, if μ is not a power of an irreducible, we conclude from Proposition 4.14.(a). In step 2.4, if $\deg(\mu) = d$ and μ is a power of an irreducible we conclude from Proposition 4.15.(a).

If the *First Loop* completes without returning an answer, we know that J is radical, and now examine the two cases.

First we look at the case when K is finite. As in the case of Algorithm 4.16, steps 3.1 and 3.2 guarantee the correctness and termination.

Now we look at the case when K is infinite. Since J is radical, checking that I is primary is equivalent to checking that J is maximal. Now, step 4 does exactly the same thing as step 5 of Algorithm 4.16 and the proof of the correctness is the same. Finally, the termination follows heuristically from Proposition 4.2.(a). □

Remark 4.20. Much as we observed in Remark 4.10, the computation of $\dim_K(P/J)$ in step IsPrimary0Dim-2.5.3 can safely be skipped if it is too costly.

Remark 4.21. When K is infinite, to turn this heuristically terminating algorithm into a true algorithm we can repeat the observations contained in Remark 4.17.

Remark 4.22. When K is finite, it would suffice to do simply steps IsPrimary0Dim-3.1 and IsPrimary0Dim-3.2 and conclude. However, our experiments suggest that nonetheless it is often faster to perform the *First Loop*, as it is quick and frequently determines the result.

Here is an example which shows that the property of being primary depends strongly on the base field.

Example 4.23. Let K be a field, let $P = K[x]$, let $f(x) = x^4 - 10\,x^2 + 1$ be the minimal polynomial of $\sqrt{2} + \sqrt{3}$, and let $I = \langle f(x) \rangle$. Now, if $K = \mathbb{Q}$, we can easily check that $f(x)$ is irreducible, hence we deduce that I is a maximal ideal. Conversely, if $K = \mathbb{F}_p$, it is known that $f(x)$ is reducible for every prime p, and hence I is not a primary ideal.

4.5. Primary Decomposition for a Zero-Dimensional Ideal

The theoretical background we shall use for computing primary decompositions of zero-dimensional ideals in affine K-algebras is explained in Chapter 5 in Kreuzer and Robbiano (2016). The main aim of this approach is to exploit our efficient algorithms for computing minimal polynomials. Here we describe the algorithms implemented in CoCoA. In particular, we remark that the algorithms for characteristic 0 (or large positive characteristic) and for finite characteristic have the same structure except for the choice of a partially splitting polynomial.

First we show how the partially splitting polynomial is chosen. The function looking for a splitting polynomial has a *First Loop* over the indeterminates; if no splitting polynomial was found, it then calls the characteristic-dependent algorithm.

In particular, if the input ideal I is primary, it returns a polynomial f such that $\mu_{f,I}$ is a power of a single irreducible factor (together with the token `TotalSplit`). Otherwise it returns a polynomial f such that $\mu_{f,I}$ has at least two irreducible factors (together with the token `PartialSplit`).

The "strange-looking" values we return in step PDSplittingFiniteField-2 and step PDSplitting-4-yes just emphasize that the ideal I is primary.

The three following functions reflect the implementation in CoCoA.

```
Algorithm 4.24. PDSplitting
notation: $P = K[x_1, \ldots, x_n]$
Input  $I$, a zero-dimensional ideal in $P$
1 compute $d = \dim_K(P/I)$
2 First Loop: for $i = 1, \ldots, n$ do
    2.1 compute $\mu_i = \mu_{x_i,I}$
    2.2 factorize $\mu_i = \prod_j^s \mu_{ij}^{d_j}$
    2.3 if $\deg(\mu_i) = d$ then return $(x_i,\ \{\mu_{ij}^{d_j} \mid j = 1, \ldots, s\},$ TotalSplit$)$
    2.4 if $s > 1$ then return $(x_i,\ \{\mu_{ij}^{d_j} \mid j = 1, \ldots, s\},$ PartialSplit$)$
3 if $K$ is finite, return PDSplittingFiniteField$(I)$
4 IsPrimary0Dim$(I)$?
    4-yes return $(0,\ \{z\},$ TotalSplit$)$
    4-no return PDSplittingInfiniteField$(I)$
Output  $(f,$ factorization of $\mu_{f,I},$ TotalSplit/PartialSplit$)$
```

Remark 4.25. In step PDSplitting-4 it would be more natural to check directly IsMaximal($\sqrt{I}$), since we have already computed all the μ_i we have practically "for free" $\sqrt{I} = I + \langle \mathrm{sqfree}(\mu_i) \mid i = 1, \ldots, n\rangle$; but calling IsMaximal entails a potentially costly computation of a Gröbner basis for $\sqrt{I}$. In our tests IsPrimary0Dim(I) was frequently significantly quicker. For instance, this is the case for Example 3.36.

The following algorithm makes a good use of $\mathrm{Frob}_q(P/I)$, the Frobenius space of P/I. Inspired by Gao et al. (2008), the detailed theoretical and computational aspects related to this concept are described in Kreuzer and Robbiano (2016), Section 5.2.

```
Algorithm 4.26. PDSplittingFiniteField
notation: $P = K[x_1, \ldots, x_n]$, $K$ a finite field
Input  $I$, a zero-dimensional ideal in $P$
1 compute FrB a $K$-basis of $\mathrm{Frob}_q(P/I)$ and let $s = \#(\mathrm{FrB})$
2 if $s = 1$ then return $(0,\ \{z\},$ TotalSplit$)$
```

```
3 pick a non-constant element $f$ of the basis FrB
4 compute $\mu = \mu_{f,I}$
5 factorize $\mu = \prod \mu_j$
6 if $\deg(\mu) = s$ then return $(f,\ \{\mu_j \mid j = 1, \ldots, s\}$, TotalSplit)
7 return $(f,\ \{\mu_j \mid j = 1, \ldots, s\}$, PartialSplit)
Output  ($f$, factorization of $\mu_{f,I}$, TotalSplit/PartialSplit)
```

Remark 4.27. From Theorem 5.2.4 in Kreuzer and Robbiano (2016) we know that for any zero-dimensional ideal I, $f \in \mathrm{Frob}_q(P/I)$ if and only if $\mu_{f,I}$ factorizes into distinct linear factors with multiplicity 1.

Algorithm 4.28. PDSplittingInfiniteField

```
notation: $P = K[x_1, \ldots, x_n]$, $K$ an infinite field
Input  $I$, a non-primary, zero-dimensional ideal in $P$
1 compute $d = \dim_K(P/I)$
2 Main Loop: repeat:
    2.1 pick a random linear form $\ell \in P$;
    2.2 compute $\mu = \mu_{\ell,I}$
    2.3 factorize $\mu = \prod_j^s \mu_j^{d_j}$
    2.4 if $\deg(\mu) = d$ then return $(\ell,\ \{\mu_j^{d_j} \mid j = 1, \ldots, s\}$, TotalSplit)
    2.5 if $s > 1$ then return $(\ell,\ \{\mu_j^{d_j} \mid j = 1, \ldots, s\}$, PartialSplit)
Output  ($\ell$, factorization of $\mu_{\ell,I}$, TotalSplit/PartialSplit)
```

Now we are ready to see how the splittings are used to compute the primary decomposition.

Algorithm 4.29. PrimaryDecompositionCore

```
notation: $P = K[x_1, \ldots, x_n]$
Input  $I$, a zero-dimensional ideal in $P$
1 let $(f,\ \{\mu_j^{d_j} \mid j{=}1,\ldots,s\}$, TotalSplit/PartialSplit) be the output of PDSplitting($I$)
2 if $s = 1$ then return $(\{I\}$, TotalSplit)
3 else return $(\{I{+}\langle \mu_j(f)^{d_j} \rangle \mid j{=}1,\ldots,s\}$, TotalSplit/PartialSplit)
Output  $(\{J_1, \ldots, J_s\}$, TotalSplit/PartialSplit)    such that $I = J_1 \cap \cdots \cap J_s$
```

Algorithm 4.30. PrimaryDecomposition0Dim

```
notation: $P = K[x_1, \ldots, x_n]$
Input  $I$, a zero-dimensional ideal in $P$
1 let $(\{J_1, \ldots, J_s\}$, TotalSplit/PartialSplit)
    be the output of PrimaryDecompositionCore($I$)
```

```
2 if it is TotalSplit, return $\{J_1, \ldots, J_s\}$
3 Main Loop: for $i = 1, \ldots, s$ do
    3.1 is $J_i$ primary?
        3.1-yes $Dec_i = \{J_i\}$
        3.1-no $Dec_i = $ PrimaryDecomposition0Dim$(J_i)$    ⟵ recursive call
4 return $Dec_1 \cup \cdots \cup Dec_s$
Output the primary decomposition of $I$
```

The column **Example** gives the reference number to the examples listed above. The other columns give, respectively, the timings (in seconds) of the computation of the algorithms 4.7, 4.16, 4.19, 4.8, and 4.30, and an indication of their answers.

Table 3. Using minimal polynomials – prime field

Example	**IsRadical**		**IsMaximal**		**IsPrimary**		**Radical**	**Primary Dec.**	
									#Comp
2.16	2.51	false	2.50	false	2.52	false	13.70	2.53	5
2.17	0.02	true	0.00	false	0.00	false	0.02	1.12	144
2.18	4.37	false	6.63	false	5.95	false	23.60	6.01	8
2.19	0.64	false	0.48	false	0.68	false	3.99	3.84	6
2.20	0.01	true	16.19	true	15.90	true	0.01	16.10	1

Table 4. Using minimal polynomials – rationals

Example	**IsRadical**		**IsMaximal**		**IsPrimary**		**Radical**	**Primary Dec.**	
									#Comp
3.28	18.08	false	4.67	false	4.30	false	22.91	54.33	2
3.29	0.46	false	0.36	false	0.36	false	0.49	3.30	2
3.30	0.87	false	0.70	false	0.46	false	1.55	1.11	2
3.31	0.01	true	1.38	true	2.61	true	0.01	1.95	1
3.32	0.03	true	> 600		> 600		0.03	> 600	
3.33	0.44	true	0.43	true	0.44	true	0.41	0.44	1
3.34	11.65	true	12.66	false	12.30	false	12.68	11.36	2
3.35	0.28	true	0.27	false	0.28	false	0.28	0.25	2
3.36	0.14	false	0.17	false	1.08	true	1.08	4.47	1

Remark 4.31. For the example 3.32 we obtained minimal polynomials which are hard to factorize (like the Swinnerton-Dyer polynomials): they have many low degree factors modulo every prime we tried. The long computation times were due to the factorizer in CoCoA.

4.6. Comparison with Singular

We present in Table 5 comparative timings of our implementation with Singular (Decker et al., 2019); in Singular we used the functions `radical` and `primdecGTZ`. It is clear that our implementation is usefully faster (and more reliable) in most cases; an exception is the primary decomposition of example 3.34 where the actual computation of the unverified result is fast, but the final verification takes more than 90% of the total time (as shown in Table 2). We had hoped to include also a comparison with Macaulay2 (Grayson and Stillman, 2019), but were unable to get timings for most of the examples.

Table 5. Singular and CoCoA– time comparisons

Example	**Radical**		**Primary Dec.**	
prime field	Singular	CoCoA	Singular	CoCoA
2.16	>600	13.70	(1) 7.87	2.53
2.17	0.01	0.02	1.00	1.12
2.18	`p too large`	23.60	`p too large`	6.01
2.19	4.91	3.99	1.62	3.84
2.20	0.01	0.01	98.15	16.10
Example	**Radical**		**Primary Dec.**	
rationals	Singular	CoCoA	Singular	CoCoA
3.28	>600	22.91	151.48	54.33
3.29	2.17	0.49	7.24	3.30
3.30	40.78	1.55	16.41	1.11
3.31	0.01	0.01	crash	1.95
3.32	0.01	0.03	crash	> 600
3.33	>600	0.41	(2) 3.62	0.44
3.34	116.29	12.68	3.82	11.36
3.35	0.86	0.28	0.11	0.25
3.36	93.83	1.08	>600	4.47

(1) `possible overflow` (2) `overflow warning`

5. Conclusion and future work

We have presented both theoretical and practical aspects of our implementations in CoCoALib for computing minimal polynomials (over $\mathbb{F}_p$). Then we presented an algorithm for computing minimal polynomials over $\mathbb{Q}$ based on a modular approach and which guarantees correctness of the result. Finally we described several algorithms which use minimal polynomials for various operations on zero-dimensional ideals (*e.g.* testing if an ideal is radical, primary or maximal).

Our experiments have shown the potential of a good implementation, and how this opens the way to new applications. For example in Abbott et al. (2018a), we use our primary decomposition approach for factoring polynomials over algebraic field extensions in advanced methods in the context of the SC^2 community: it is proving to be useful in the software CArL/SMT-RAT by Kremer and Ábrahám (2018) which implements Lazard's variant of Cylindrical Algebraic Decomposition.

On the theoretical side, we investigate more deeply the consequences of our new modular approach and apply it to general ideals in the preprint by Abbott et al. (2018b).

References

Abbott, J., 2012. Twin-float arithmetic. J. Symb. Comput. 47 (5), 536–551.

Abbott, J., 2017. Fault-Tolerant Modular Reconstruction of Rational Numbers. J. Symb. Comp. 80, 707–718.

Abbott, J., Bigatti, A., 2017. New in CoCoA-5.2.2 and CoCoALib-0.99560 for SC-Square. In: Proceedings of the 2nd International Workshop on Satisfiability Checking and Symbolic Computation. Vol. 1974. pp. 1–6.

Abbott, J., Bigatti, A., 2019. CoCoALib: a C++ library for doing Computations in Commutative Algebra. Available at `http://cocoa.dima.unige.it/cocoalib`.

Abbott, J., Bigatti, A., Palezzato, E., 2018a. New in CoCoA-5.2.4 and CoCoALib-0.99600 for SC-Square. In: Proceedings of the 3rd Workshop on Satisfiability Checking and Symbolic Computation. Vol. 2189. pp. 88–94.

Abbott, J., Bigatti, A., Robbiano, L., 2017. Implicitization of Hypersurfaces. J. Symb. Comput. 81, 20–40.

Abbott, J., Bigatti, A., Robbiano, L., 2018b. Ideals modulo *p*. `ArXiv:1801.06112`.

Abbott, J., Bigatti, A., Robbiano, L., 2019. CoCoA: a system for doing Computations in Commutative Algebra. Available at `http://cocoa.dima.unige.it`.

Aoyama, T., Noro, M., 2018. Modular Algorithms for Computing Minimal Associated Primes and Radicals of Polynomial Ideals. In: Proc. ISSAC 2018. pp. 31–38.

Arnold, E., 2003. Modular algorithms for computing Gröbner bases. J. Symb. Comput. 35, 403–419.

Bostan, A., Salvy, B., Schost, E., 2003. Fast Algorithms for Zero-Dimensional Polynomial Systems Using Duality. AAECC 14, 239–272.

Buchberger, B., 1985. Gröbner Bases: An Algorithmic Method in Polynomial Ideal Theory.

Collins, G., 1975. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In: Automata Theory and Formal Languages 2nd GI Conference. pp. 134–183.

Decker, W., Greuel, G.-M., Pfister, G., Schönemann, H., 2019. SINGULAR, *A computer algebra system for polynomial computations*. Available at `http://www.singular.uni-kl.de/`.

Faugère, J., Gianni, P., Lazard, D., Mora, T., 1993. Efficient Computation of Zero-dimensional Gröbner Bases by Change of Ordering. J. Symb. Comput. 16, 329–344.

Gao, S., Wan, D., Wang, M., 2008. Primary Decomposition of Zero-Dimensional Ideals over Finite Fields. Math. Comp. 78, 509–521.

Gräbe, H.-G., 1988. On Lucky Primes. J. Symb. Comput. 6, 183–208.

Grayson, D., Stillman, M., 2019. Macaulay2, a software system for research in algebraic geometry. Available at `http://www.math.uiuc.ed/Macaulay2/`.

Idrees, N., Pfister, G., Steidel, S., 2011. Parallelization of Modular Algorithms. J. Symb. Comput. 46, 672–684.

Kremer, G., Ábrahám, E., 2018. Modular strategic SMT solving with SMT-RAT.

Kreuzer, M., Robbiano, L., 2008. Computational Commutative Algebra 1, 2nd Edition. Springer, Heidelberg.

Kreuzer, M., Robbiano, L., 2016. Computational Linear and Commutative Algebra. Springer, Heidelberg.

Lazard, D., 1992. Solving zero-dimensional algebraic systems. J. Symb. Comput. 13, 117–133.

McCallum, S., Parusiński, A., Paunescu, L., 2017. Validity proof of Lazard's method for CAD construction. J. Symb. Comput. 92, 52–69.

Monagan, M., 2004. Maximal Quotient Rational Reconstruction: An Almost Optimal Algorithm for Rational Reconstruction. In: Proc. ISSAC, ACM 2004. pp. 243–249.

Mora, T., Robbiano, L., 1993. The Gröbner Fan of an Ideal. J. Symb. Comput. 15, 199–209.

Noro, M., 2002. An efficient modular algorithm for computing the global b-function. In: Proceeding of the First Congress of Mathematical Software, A.M. Cohen, X.S. Gao, N. Takayama, eds. pp. 147–157.

Noro, M., Yokoyama, K., 1999. A modular method to compute the rational univariate representation of zero-dimensional ideals. J. Symb. Comput. 28, 243–263.

Noro, M., Yokoyama, K., 2004. Implementation of prime decomposition of polynomial ideals over small finite fields. J. Symb. Comput. 38, 1227–1246.

Noro, M., Yokoyama, K., 2018. Usage of Modular Techniques for Efficient Computation of Ideal Operations. Math.Comput.Sci. 12, 1–32.

Pauer, F., 2007. Gröbner bases with coefficients in rings. J. Symb. Comput. 42, 1003–1011.

Rouillier, F., 1999. Solving Zero-Dimensional Systems Through the Rational Univariate Representation. AAECC 9, 433–461.

Winkler, F., 1988. A p-adic Approach to the Computation of Gröbner Bases. J. Symb. Comput. 6, 287–304.