A Signature-based Algorithm for Computing the Nondegenerate Locus of a Polynomial System

Christian Eder, Pierre Lairez, Rafael Mohr, Mohab Safey El Din

A Signature-based Algorithm for Computing the Nondegenerate Locus of a Polynomial System

Christian Eder * Pierre Lairez † Rafael Mohr *‡ Mohab Safey El Din ‡

February 28, 2022

Abstract

Polynomial system solving arises in many application areas to model non-linear geometric properties. In such settings, polynomial systems may come with degeneration which the end-user wants to exclude from the solution set. The nondegenerate locus of a polynomial system is the set of points where the codimension of the solution set matches the number of equations.

Computing the nondegenerate locus is classically done through ideal-theoretic operations in commutative algebra such as saturation ideals or equidimensional decompositions to extract the component of maximal codimension.

By exploiting the algebraic features of signature-based Gröbner basis algorithms we design an algorithm which computes a Gröbner basis of the equations describing the closure of the nondegenerate locus of a polynomial system, without computing first a Gröbner basis for the whole polynomial system.

1 Introduction

Problem Statement Fix a field $\mathbb{K}$ with an algebraic closure $\overline{\mathbb{K}}$ and a polynomial ring $R := \mathbb{K}[x_1, \dots, x_n]$ over $\mathbb{K}$. Let $f_1, \dots, f_c \in R$ with $c \leq n$ and $V := \left\{p \in \overline{\mathbb{K}}^n \;\middle|\; f_1(p) = \cdots = f_c(p) = 0\right\}$. This algebraic set is a finite union of irreducible components. By the Principal Ideal Theorem [20, Theorem 10.2] the codimension of the $\mathbb{K}$-irreducible components of V is at most c. Let V_c denote the union of the components of V of codimension exactly c.

The goal of this paper is to compute a Gröbner basis of an ideal whose zero set is V_c, which we call the *nondegenerate part* of the system $f_1, \dots, f_c$ (or rather *a* nondegenerate part, since this is unique only up to radicality).

Prior works and scientific locks. The state-of-the-art algorithms to compute the nondegenerate part of I rely on the more general problem of computing the equidimensional decomposition of I. There is a vast body of literature split along what data structure is used for the output into two research lines.

The first family of algorithms computes a Gröbner basis of the ideal of each component. There are two different approaches in this line. The first uses projections, computed with *elimination orderings*, to reduce the problem to a problem for hypersurfaces [7; 25; 32]. The second relies on homological characterizations of the dimension and the computation of free resolutions [21]. See Decker et al. [14] and Greuel and Pfister [29] for further references. Both approaches use Gröbner basis algorithms as a black box for performing various ideal-theoretic operations, in particular ideal quotients (also known as colon ideals).

*TU Kaiserslautern, Germany
†Inria, Uni. Paris-Saclay, Palaiseau, France
‡Sorbonne Uni., CNRS, Paris, France

A second family of algorithms outputs equidimensional components of I or its radical through *lazy* representations, i.e. as complete intersections over a non-empty Zariski open set. This is the case for the so-called regular chains which go back to Wu-Ritt characteristic sets [43].

These put into practice a kind of D5 principle [17] to split geometric objects by enforcing an equiprojectability property. See [1; 11; 30; 35; 41; 42] for further references. When the base field k has characteristic 0 (or large enough characteristic), geometric resolution algorithms [26] can also be used. These culminate with the incremental algorithm in [33; 34] which avoids equiprojectability issues by performing a linear change of variables to ensure Noether position properties. One feature is that input polynomials are encoded with straight-line programs to take advantage of evaluation properties. See also [31] for a similar approach. It also gives the best known complexity for equidimensional decomposition: linear in the evaluation complexity of the input system and polynomial in some algebraic degree.

On the software side, the computer algebra systems Singular [16], Macaulay2 [28] and Magma [5] implement the algorithm of Eisenbud et al. [21] to perform equidimensional decomposition. Maple implements algorithms for computing regular chains [8; 9; 12; 13; 36] and algorithms based on Gröbner bases. The algorithm by Gianni et al. [25] is used for prime decomposition and, combined with techniques from [2], for equidimensional decomposition. All these implementations use Gröbner basis algorithms as a black box.

Main results. As opposed to previous work, we only focus on computing the nondegenerate part of an ideal, not the full equidimensional decomposition. Introducing splitting techniques in the middle of Gröbner basis algorithms is a natural idea [e.g. 27]. A topical issue is then to control the computational cost of these splittings. This raises two scientific locks: *(i)* when should one proceed with a splitting attempt and *(ii)* how to minimize the cost of each splitting by exploiting algebraic knowledge of previous computations.

In this paper, we do not provide any complexity result but address *(i)* and *(ii)*. Problem *(i)* is solved, following an old idea by Faugère [24], by revisiting the idea of using syzygies already present in [21] through the prism of signature-based Gröbner basis algorithms such as the F5 algorithm [18; 23].

In order to compute a Gröbner basis of the ideal $\langle f_1, \ldots, f_c \rangle$, the F5 algorithm computes Gröbner bases of $\langle f_1, \ldots, f_i \rangle$ incrementally "à la Buchberger" but keeps track of an auxiliary data structure, called a *signature*, which is attached to each considered polynomial. This enables one to avoid certain reductions to zero in Buchberger's algorithm, namely the ones coming from trivial *a priori* syzygies.

Concretely, in the F5 algorithm, a reduction to zero happens exactly when some f_{i+1} is a zero divisor in $R/\langle f_1, \ldots, f_i \rangle$ [18, Corollary 7.2]. This gives a hint on *when* to apply a splitting procedure. We show *how* to apply it through a procedure with successive saturations of polynomial ideals to mimic in algebraic terms the set theoretic operations which consist in "computing" the intersection of $V(f_{i+1})$ with the Zariski closure of $V(f_1, \ldots, f_i) - V(f_{i+1})$. We then show how to remove superflous irreducible components from the result via a "cleanup" step. This solves Problem *(i)*.

To tackle *(ii)*, we further apply the properties of signatures which are maintained during the run of the F5 algorithm. These enable one to keep track of crucial information of the syzygy module of the ideal under consideration but in a lazy and compact way, which minimizes the impact of this extra data structure to practical computations. It turns out that these signatures encompass enough information to efficiently perform the saturation and "cleanup" operations we need in the splitting procedure designed to solve *(i)*.

All in all, we then show how to modify the F5 algorithm to successively compute the highest codimension components of each intermediate ideal generated by the first i equations.

We implemented our algorithm using the programming language Julia [4]. We illustrate its efficiency by first comparing it with our implementation of the F5 algorithm. Practical experiments show that the overhead induced by our splitting techniques is significantly reduced by introducing signature-based techniques into the procedure designe to solve *(i)*. We also show that our implementation tackle examples which are out of reach to current implementations in systems such as Macaulay2, Maple and Singular.

2 Auxiliary results

We refer the reader to e.g. [20] for further details on the notions of primary decomposition and dimension used in this section.

Let I be an ideal of R as above.

Definition 2.1. *The set of minimal associated primes of I, denoted* $\mathrm{MinAss}(I)$*, is the set of prime ideals of R which contain I and are minimal w.r.t. inclusion. The algebraic sets associated to the minimal associated primes of I are the irreducible components of $V(I)$.*

Let $\mathrm{MinAss}_c(I)$ *denote the set of minimal associated primes of I whose codimension is eactly c, We define the locus of codimension c of I as $I_c := \bigcap_{\mathfrak{p}\in \mathrm{MinAss}_c(I)} \mathfrak{p}$ (and $\langle 1\rangle$ if* $\mathrm{MinAss}_c(I)$ *is empty). We say that I is pure of codimension c if $\sqrt{I} = I_c$. If $I = \langle f_1, \ldots, f_c\rangle$ is generated by c elements then any ideal J satisfying $\sqrt{J} = I_c$ is called the nondegenerate locus of $f_1, \ldots, f_c$.*

For two ideals $J, K \subseteq R$ we write $J \overset{\mathrm{rad}}{=} K$ if $\sqrt{J} = \sqrt{K}$, i.e. if $V(I) = V(J)$.

Assume for the remainder of this section that we are given $f_1, \ldots, f_c \in R$ with $c \leq n$ and denote $I := \langle f_1, \ldots, f_c\rangle$. If $c > n$ then the nondegenerate locus of $f_1, \ldots, f_c$ is trivial so we only treat the case $c \leq n$ here. Any minimal associated prime of I has at most codimension c by the Principal Ideal Theorem [20, Chap. 10].

The goal of this section is to obtain an algorithm which computes generators for the nondegenerate locus J of $f_1, \ldots, f_c$. For the moment, this algorithm will be described using the ideal theoretic operation of computing the generators of the *saturation* of one ideal by another. Let us recall the definition:

Definition 2.2. *Let $J, K \subseteq R$ be two ideals. The colon ideal of J w.r.t. K is defined as $(J : K) := \{p \in R \mid pK \subseteq J\}$ where $pK := \{pf \mid f \in K\}$. The saturation of J w.r.t. K is defined as*

$$(J : K^\infty) = \left\{p \in R \;\middle|\; \exists k \in \mathbb{N} \text{ such that } pK^k \subseteq J\right\}.$$

To obtain our algorithm, we gather some well known facts about colon ideals and saturations.

Proposition 2.3. *Let $J, K \subseteq R$ be two ideals and let $J = \bigcap_{i=1}^r Q_i$ be a minimal primary decomposition.*

1. *We have*

$$(J : K^\infty) = \bigcap_{K \not\subseteq \sqrt{Q_i}} Q_i.$$

   In particular $\mathrm{MinAss}((J : K^\infty)) \subseteq \mathrm{MinAss}(J)$.

2. *If K is equal to the intersection of some of the Q_i's, say $K = \bigcap_{i=1}^s Q_i$ for some $s \leq r$, then*

$$(J : K) \overset{\mathrm{rad}}{=} \bigcap_{i=s+1}^{r} Q_i.$$

Proof. For statement (1), see e.g. [21, Lemma 2.4]. To prove statement (2), note that if $p \in (J : K)$ then, for any of the Q_i's, we have, by the definition of a primary ideal, that either $p \in Q_i$ or $K \subset Q_i$ or both $p \in \sqrt{Q_i}$ and $K \subset \sqrt{Q_i}$. Since the chosen primary decomposition is minimal we hence must have $p \in \sqrt{Q_i}$ for all $i \geq s + 1$. The reverse inclusion is obvious. □

The cornerstone of our algorithm will be the following statement.

Lemma 2.4. *For any ideal $J \subseteq R$ and any $f \in R$ we have*

$$J + \langle f \rangle \overset{\text{rad}}{=} ((J : f^\infty) + \langle f \rangle) \cap (J : (J : f^\infty))$$

If J is pure of codimension $c < n$ then $((J : f^\infty) + \langle f \rangle)$ is pure of codimension $c+1$ and $(J : (J : f^\infty))$ is pure of codimension c.

Remark 2.1. Geometrically, by Proposition 2.3, the irreducible components of $V((J : f^\infty))$ are the irreducible components of $V(J)$ which are transversely intersected by f and the irreducible components of $V((J : (J : f^\infty)))$ are the irreducible components of $V(J)$ on which f is identically zero.

Proof. Let $p \in J + \langle f \rangle$. We prove that p is in the radical of $(J : f^\infty) + \langle f \rangle$ and that of $(J : (J : f^\infty))$. The former inclusion follows simply from $J \subseteq (J : f^\infty)$. Concerning the latter, let $g \in (J : f^\infty)$, so that $gf^\ell \in J$ for some $\ell \in \mathbb{N}$. We write $p = q + af$ where $q \in J$ and $a \in R$. Then $p^\ell = q' + a'f^\ell$ for some $q' \in J$, $a' \in R$ and so

$$gp^\ell = gq' + a'gf^\ell \in J.$$

Therefore $p \in \sqrt{J : (J : f^\infty)}$.

Conversely, let $p \in ((J : f^\infty) + \langle f \rangle) \cap (J : (J : f^\infty))$. Write $p = q + af$ where $q \in (J : f^\infty)$ and $a \in R$. Since $p \in (J : (J : f^\infty))$, we have $pq \in J$. But $pq = q^2 + aqf$, so $q^2 \in J + \langle f \rangle$. It follows that $q \in \sqrt{J + \langle f \rangle}$, thus proving the stated equality.

To prove that the ideals on the right hand side are pure of codimension $c+1$ respectively c, note that if $(J : f^\infty)$ is contained in some primary component P of J and if Q is another primary components of J with $\sqrt{P} \subseteq \sqrt{Q}$ then also $(J : f^\infty) \subseteq Q$. By Proposition 2.3 we hence have $\text{MinAss}\,((J : (J : f^\infty))) \subseteq \text{MinAss}(J)$. Hence $(J : (J : f^\infty))$ is pure of codimension c. Also by Proposition 2.3 and the prime avoidance lemma there exists $g \in (J : f^\infty)$ such that $f + g$ is not contained in any minimal associated prime of J and hence is not a zero divisor modulo J. Hence there exists a regular sequence of length $c+1$ in $(J : f^\infty) + \langle f \rangle$. The equidimensionality of $(J : f^\infty) + \langle f \rangle$ then follows from the Principal Ideal Theorem. □

We are now ready to describe Algorithm 1. To do this we suppose for now that we have an algorithm `sat` which given a set of generators for an ideal J and a set of generators for an ideal K returns a set of generators for $(J : K^\infty)$ and an analogous algorithm `colon` returning generators for $(J : K)$. Given $c \leq n$ elements $f_1, \ldots, f_c \in R$ the core loop of Algorithm 1 is now to start with the ideal $J = \langle f_1 \rangle$ and to continously replace it with $(J : f_k^\infty) + f_k$ for each k. By Lemma 2.4 the resulting ideal will be equidimensional of codimension c. Note however that it may have minimal associated primes that the original ideal $I = \langle f_1, \ldots, f_c \rangle$ does not have:

Example 2.5. *Let $R = k[x, y, z]$ and $f_1 = xy$, $f_2 = xz$. Then $(xy : xz^\infty) + xz = \langle y, xz \rangle$ which has the minimal associated prime $\langle y, x \rangle$ which is not a minimal associated prime of $\langle f_1, f_2 \rangle = \langle x \rangle \cap \langle y, z \rangle$.*

These additional components are removed at every iterative step with an additional saturation in lines 6 through 8 of Algorithm 1.

Proposition 2.6. *Let $J, K, L \subseteq R$ be ideals in R such that $J \overset{\text{rad}}{=} K \cap L$. Then we have for any $f \in R$*

$$J + \langle f \rangle \overset{\text{rad}}{=} (K + \langle f \rangle) \cap (L + \langle f \rangle)$$

Proof. We have

$$\begin{aligned}(K + \langle f \rangle) \cap (L + \langle f \rangle) &\overset{\text{rad}}{=} (K + \langle f \rangle)(L + \langle f \rangle) \\ &\overset{\text{rad}}{=} KL + \langle f \rangle \overset{\text{rad}}{=} (K \cap L) + \langle f \rangle \\ &\overset{\text{rad}}{=} J + \langle f \rangle.\end{aligned}$$

□

Algorithm 1 Computation of the nondegenerate part

Input: A set of generators $f_1, \dots, f_c$ for an ideal I in R where $c \leq n$
Output: A set of generators G for the nondegenerate part of $f_1, \dots, f_c$

1: $G \leftarrow \{f_1\}$
2: $\mathcal{K} \leftarrow \emptyset$
3: **for** $k \in \{2, \dots, c\}$ **do**
4: $\quad G \leftarrow \mathtt{sat}(G, f_k) \cup \{f_k\}$
5: $\quad \mathcal{K} \leftarrow \mathcal{K} \cup \{\mathtt{colon}(G, \mathtt{sat}(G, f_k))\}$
6: $\quad$ **for** $K \in \mathcal{K}$ **do**
7: $\quad\quad G \leftarrow \mathtt{sat}(G, K)$
8: $\quad$ **end for**
9: **end for**
10: **return** G

Theorem 2.7. *Algorithm 1 is correct.*

Proof. Let c' be minimal such that

$$I' := \langle f_1, \dots, f_{c'} \rangle \neq (\langle f_1, \dots, f_{c'-1} \rangle : f_{c'}^\infty) + \langle f_{c'} \rangle.$$

By Lemma 2.4 and Proposition 2.6 we have

$$I \overset{\text{rad}}{=} \underbrace{((I' : f_{c'}^\infty) + \langle f_{c'}, \dots, f_c \rangle)}_{=:I_1} \cap \underbrace{((I' : (I' : f_{c'}^\infty)) + \langle f_{c'+1}, \dots, f_c \rangle)}_{=:I_2}$$

and $\operatorname{codim} I_2 < \operatorname{codim} I_1 \leq c$. Therefore by Proposition 2.3

$$(I_1 : I_2^\infty) = (I_1 : (I' : (I' : f_{c'}^\infty))^\infty)$$

has all minimal associated primes of codimension c of I as minimal associated primes. The claim now easily follows by induction on the number of times $\langle G \rangle \neq (\langle G \rangle : f_k^\infty)$ in line 4 of Algorithm 1. □

3 The F5 Algorithm

Our presentation of the F5 algorithm in the following closely follows [18]. While Algorithm 3, which we call F5, is slightly different from the original presentation of F5 in [23] it still follows the original F5 very closely. See [18, section 8] for a detailed description of the exact differences between the original F5 and Algorithm 3.

In the following fix an ordered finite sequence $(f_1, \dots, f_r) \subset R$. A *term* in R will be a polynomial of the form cm where $c \in \mathbb{K}$ and m is a monomial. We also fix a *monomial ordering* $\leq_R$ on the set of monomials in R, i.e. a total ordering on the set of monomials in R extending the partial ordering given by divisibility.

Definition 3.1. *If $f \in R$ can be written as $f = \sum_{k=1}^s c_k m_k$ with linearly independent terms $c_k m_k$, $k = 1, \dots, s$, then the leading monomial of f, denoted $\operatorname{lm}(f)$, is defined as the $\leq_R$-maximal element in $\{m_1, \dots, m_k\}$. If $\operatorname{lm}(f) := m_\ell$ then the leading term of f, denoted $\operatorname{lt} f$, is defined as $\operatorname{lt}(f) := c_\ell m_\ell$.*

Denote by R^r the free module of rank r over R and by $\mathbf{e}_i \in R^r$ the ith standard unit vector. We denote

by $\overline{\bullet} : R^r \to R$ the morphism

$$\overline{\bullet} : R^r \to R$$
$$\boldsymbol{\alpha} = \sum_{i=1}^{r} \alpha_i \mathbf{e}_i \mapsto \overline{\boldsymbol{\alpha}} := \sum_{i=1}^{r} \alpha_i f_i.$$

For a set $G \subset R^r$ denote by $\overline{G}$ the image of G under $\overline{\bullet}$.

Definition 3.2. *An element in the kernel of $\overline{\bullet}$ is called a syzygy of $(f_1, \ldots, f_r)$. For two elements $\boldsymbol{\alpha}, \boldsymbol{\beta} \in R^r$ we define the Koszul syzygy of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ by $K(\boldsymbol{\alpha}, \boldsymbol{\beta}) := \overline{\boldsymbol{\beta}}\boldsymbol{\alpha} - \overline{\boldsymbol{\alpha}}\boldsymbol{\beta} \in R^r$.*

We extend the notions of monomials and leading monomials to R^r:

Definition 3.3. *A module monomial is an element $m\mathbf{e}_i \in R^r$ where m is a monomial in R. The position over term ordering on the set of module monomials in R^r is defined by*

$$m\mathbf{e}_i <_{POT} n\mathbf{e}_j \quad \Leftrightarrow \quad \begin{array}{l} i < j \text{ or} \\ i = j \text{ and } m <_R n. \end{array}$$

The signature of an element $\alpha \in R^r$, denoted $\mathfrak{s}(\boldsymbol{\alpha})$, is the $\leq_{POT}$-maximal module monomial occuring in $\boldsymbol{\alpha}$. A syzygy signature is the signature of a syzygy of $(f_1, \ldots, f_r)$. If $\mathfrak{s}(\boldsymbol{\alpha}) = m\mathbf{e}_i$ then we define the index of $\boldsymbol{\alpha}$ as $\mathrm{ind}(\boldsymbol{\alpha}) := i$. We also say that i is the index corresponding to f_i. The signature degree of $\boldsymbol{\alpha}$ is the degree of m.

When the context is clear we will denote both the ordering $\leq_R$ and $\leq_{\mathrm{POT}}$ simply by $\leq$.

The classical method to compute a Gröbner basis for an ideal is Buchberger's Algorithm. Its basic structure is to build certain polynomials, called *S-pairs*, from two given polynomials in the intermediate basis G of the output Gröbner basis, *reduce*, i.e. compute a certain multivariate remainder of, this polynomial by G and add the result of this reduction to G if it is non-zero. Because a polynomial reducing to zero is hence useless for the output of the algorithm we would like to avoid as many zero reductions as possible. In order to achieve this, the F5 Algorithm modifies the basic structure of Buchberger's algorithm in four steps:

Algorithmic Specification 3.1.

1. Replace every polynomial f occuring in Buchberger's Algorithm by an element $\boldsymbol{\alpha} \in R^r$ such that $\overline{\boldsymbol{\alpha}} = f$. A reduction of $\boldsymbol{\alpha}$ by some $\boldsymbol{\beta} \in R^r$ is then an operation $\boldsymbol{\alpha} - s\boldsymbol{\beta}$ for some term $s \in R$ such that the operation $\overline{\boldsymbol{\alpha}} - s\overline{\boldsymbol{\beta}}$ cancels some term in $\overline{\boldsymbol{\alpha}}$.

2. Allow only reductions that never change the signature of $\boldsymbol{\alpha}$, called *regular $\mathfrak{s}$-reductions*, i.e. in the notation of (1) we require that $\mathfrak{s}(m\boldsymbol{\beta}) < \mathfrak{s}(\boldsymbol{\alpha})$ whenever we perform a reduction of $\boldsymbol{\alpha}$ (see definition 3.4).

3. Process the S-pairs, which are now called $\mathfrak{s}$-pairs in the context of F5, by order of increasing signature w.r.t $\leq_{\mathrm{POT}}$.

4. It then turns out that modifications (1)-(3) yield the following two improvements over the basic Buchberger Algorithm:

   (a) Any $\mathfrak{s}$-pair whose signature is divisible by a syzygy signature is known to *a priori* reduce to zero, i.e. does not need to be reduced explicitly (see Lemma 3.8).

   (b) If there are several $\mathfrak{s}$-pairs of the same signature, all but one can be discarded (see Lemma 3.9).

Remark 3.1. It should be mentioned that in any practical implementation, if one's goal is only to compute a Gröbner basis for the ideal $\langle f_1, \ldots, f_r \rangle$ via F5, one replaces each module element $\boldsymbol{\alpha}$ by the pair $(\mathfrak{s}(\boldsymbol{\alpha}), \overline{\boldsymbol{\alpha}})$. Storing only this information turns out to be sufficient to be able to implement algorithmic specification 3.1. However, for clarity of presentation, we work with full module elements in the following description of F5.

We will now introduce these modifications in more detail.

As mentioned above, the concept which replaces the usual polynomial reduction in Buchberger's Algorithm is that of a *regular $\mathfrak{s}$-reduction*:

Definition 3.4. *Let $\boldsymbol{\alpha} \in R^r$. An $\mathfrak{s}$-reduction of $\boldsymbol{\alpha}$ by an element $\boldsymbol{\beta} \in R^r$ is the operation $\boldsymbol{\alpha} - s\boldsymbol{\beta}$ for some term s such that $s\,\mathrm{lt}(\overline{\boldsymbol{\beta}}) = t$ for some term t in $\overline{\boldsymbol{\alpha}}$ and such that $\mathfrak{s}(s\boldsymbol{\beta}) \leq \mathfrak{s}(\boldsymbol{\alpha})$. It is a top-$\mathfrak{s}$-reduction if $t = \mathrm{lt}(\overline{\boldsymbol{\alpha}})$. The $\mathfrak{s}$-reduction is called regular if $\mathfrak{s}(s\boldsymbol{\beta}) < \mathfrak{s}(\boldsymbol{\alpha})$. An $\mathfrak{s}$-reduction of $\boldsymbol{\alpha}$ by a set $G \subset R^r$ to some element $\boldsymbol{\gamma} \in R^r$ is a sequence of $\mathfrak{s}$-reductions of $\boldsymbol{\alpha}$ by elements in G that yields $\boldsymbol{\gamma}$ as a result. One says that $\boldsymbol{\alpha}$ is fully regular $\mathfrak{s}$-reduced by G if there exists no regular $\mathfrak{s}$-reducer for $\boldsymbol{\alpha}$ in G.*

The notion of Gröbner bases is translated to the notion of *signature Gröbner bases*:

Definition 3.5. *A signature Gröbner basis (of $(f_1, \ldots, f_r)$) up to signature T is a finite subset $G \subset R^r$ such that every element $\boldsymbol{\alpha} \in R^r$ with $\mathfrak{s}(\boldsymbol{\alpha}) < T$ $\mathfrak{s}$-reduces to a syzygy by G. It is a signature Gröbner basis (of $(f_1, \ldots, f_r)$) if it is a signature Gröbner basis up to every signature.*

Remark 3.2. Note that the definition of a signature Gröbner basis immediately implies that if G is a signature Gröbner basis up to some signature T and $\boldsymbol{\alpha} \in R^r$ with $\mathfrak{s}(\boldsymbol{\alpha}) = T$ then $\boldsymbol{\alpha}$ always regular $\mathfrak{s}$-reduces to the same fully regular $\mathfrak{s}$-reduced result by G no matter which reducers we choose. In this context we can thus speak of *the* result of regular $\mathfrak{s}$-reducing $\boldsymbol{\alpha}$ by G. Because any monomial ordering is a well ordering, regular $\mathfrak{s}$-reducing $\boldsymbol{\alpha}$ by G always requires only finitely many $\mathfrak{s}$-reductions of $\boldsymbol{\alpha}$ by elements in G.

Note that if $G \subset R^r$ is a signature Gröbner basis then $\overline{G} := \{\overline{\boldsymbol{\alpha}} \mid \boldsymbol{\alpha} \in G\}$ is a Gröbner basis of $\langle f_1, \ldots, f_r \rangle$ [18, Lemma 4.1]. It should also be emphasized that while the notion of a Gröbner basis is independent of any ordering of the elements $f_1, \ldots, f_r$ the notion of a *signature* Gröbner basis is very much not. This is because the signature of an element in R^r depends on the module monomial ordering $\leq_{\mathrm{POT}}$ which in turn depends on the ordering of the $f_1, \ldots, f_r$.

We replace the notion of S-pairs with the one of *regular $\mathfrak{s}$-pairs*.

Definition 3.6. *Let $\boldsymbol{\alpha}, \boldsymbol{\beta} \in R^r$. The $\mathfrak{s}$-pair of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ is*

$$\mathfrak{s}p(\boldsymbol{\alpha}, \boldsymbol{\beta}) = a\boldsymbol{\alpha} - b\boldsymbol{\beta}$$

where

$$a = \frac{\mathrm{lcm}(\mathrm{lm}(\overline{\boldsymbol{\alpha}}), \mathrm{lm}(\overline{\boldsymbol{\beta}}))}{\mathrm{lt}(\overline{\boldsymbol{\alpha}})};\ b = \frac{\mathrm{lcm}(\mathrm{lm}(\overline{\boldsymbol{\alpha}}), \mathrm{lm}(\overline{\boldsymbol{\beta}}))}{\mathrm{lt}(\overline{\boldsymbol{\beta}})}.$$

The $\mathfrak{s}$-pair is called regular if $\mathfrak{s}(a\boldsymbol{\alpha}) \neq \mathfrak{s}(b\boldsymbol{\beta})$.

We now have the following signature-analogue of Buchberger's criterion [18, Theorem 4.1]:

Theorem 3.7. *A subset $G \subset R^r$ is a signature Gröbner basis up to signature T if and only if every $\mathbf{e}_i$ with $\mathbf{e}_i < T$ and every regular $\mathfrak{s}$-pair $\mathfrak{s}p(\boldsymbol{\alpha}, \boldsymbol{\beta})$ for $\boldsymbol{\alpha}, \boldsymbol{\beta} \in G$ with $\mathfrak{s}(\mathfrak{s}p(\boldsymbol{\alpha}, \boldsymbol{\beta})) < T$ $\mathfrak{s}$-reduces to a syzygy by G.*

From this we obtain Algorithm 2 as a signature-based analogue of Buchberger's Algorithm.

Remark 3.3. The effect of the choice of the ordering $\leq_{\mathrm{POT}}$ in Algorithm 2 is that for any $1 \leq k < m$ a full signature Gröbner basis for $(f_1, \ldots, f_k)$ will be computed before any $\mathfrak{s}$-pair in index $k+1$ is considered since $\mathfrak{s}$-pairs in index k always have lower signature than $\mathfrak{s}$-pairs in index $k+1$.

Algorithm 2 Buchberger with Signatures

Input: An ordered sequence $(f_1, \ldots, f_r)$ in R
Output: A signature Gröbner basis G of $(f_1, \ldots, f_r)$

```
1: G ← ∅
2: P ← {e_1, ..., e_r}
3: while P ≠ ∅ do
4:    α ← the ≤_POT-minimal element in P
5:    P ← P \ {α}
6:    γ ← the result of fully regular 𝔰-reducing α by G
7:    if γ is not a syzygy then
8:       G ← G ∪ {γ}
9:       P ← P ∪ {𝔰p(γ, β) | β ∈ G}
10:   end if
11: end while
12: return G
```

Note that by Remark 3.2, line 8 in Algorithm 2 is a well defined operation. The fact that we compute our signature Gröbner basis by order of increasing signature will be key for implementing algorithmic specification 3.1 (4). The following two lemmas allow us to do just that [18, Lemmas 6.1 and 6.2]:

Lemma 3.8 (Syzygy criterion)**.** *Let $\boldsymbol{\alpha} \in R^r$ and let G be a signature Gröbner basis up to signature $\mathfrak{s}(\boldsymbol{\alpha})$. Suppose that there exists a syzygy signature T with $T \mid \mathfrak{s}(\boldsymbol{\alpha})$. Then $\boldsymbol{\alpha}$ $\mathfrak{s}$-reduces to a syzygy by G.*

The syzygy criterion implies that any $\mathfrak{s}$-pair whose signature is divisible by a syzygy signature may be discarded before reduction in Algorithm 2.

Lemma 3.9 (Singular criterion)**.** *Let $\boldsymbol{\alpha}, \boldsymbol{\beta} \in R^r$ and let G be a signature Gröbner basis up to signature $\mathfrak{s}(\boldsymbol{\alpha}) = \mathfrak{s}(\boldsymbol{\beta})$. Let $\boldsymbol{\gamma}_{\boldsymbol{\alpha}}$ and $\boldsymbol{\gamma}_{\boldsymbol{\beta}}$ be the results of regular $\mathfrak{s}$-reducing $\boldsymbol{\alpha}$ respectively $\boldsymbol{\beta}$ by G. Then $\overline{\boldsymbol{\gamma}_{\boldsymbol{\alpha}}} = \overline{\boldsymbol{\gamma}_{\boldsymbol{\beta}}}$.*

The singular criterion implies that out of any two $\mathfrak{s}$-pairs with the same signature one has to regular $\mathfrak{s}$-reduce only one in Algorithm 2. Combining both criteria, we see that we have to regular $\mathfrak{s}$-reduce *at most one* $\mathfrak{s}$-pair per signature. The choice of which $\mathfrak{s}$-pair then to reduce is made, for example, by the *rewrite criterion*:

Definition 3.10. *Let G be a signature Gröbner basis up to some signature T, let $\boldsymbol{\alpha} \in G$ and let $a \in R$ be a monomial. The element $a\boldsymbol{\alpha}$ with $\mathfrak{s}(a\boldsymbol{\alpha}) = T$ is said to be rewritable w.r.t. G if either*

- *there exists $\boldsymbol{\beta} \in G$ with $\mathfrak{s}(\boldsymbol{\beta}) > \mathfrak{s}(\boldsymbol{\alpha})$ and $\mathfrak{s}(\boldsymbol{\beta}) \mid \mathfrak{s}(a\boldsymbol{\alpha})$ or*
- *There exists a syzygy signature T with $T \mid \mathfrak{s}(a\boldsymbol{\alpha})$.*

Let again G' be an intermediate state of G in Algorithm 2 and let $\mathfrak{s}p(\boldsymbol{\alpha}, \boldsymbol{\beta}) = a\boldsymbol{\alpha} - b\boldsymbol{\beta}$ for $\boldsymbol{\alpha}, \boldsymbol{\beta} \in G'$. It turns out that if either $a\boldsymbol{\alpha}$ or $b\boldsymbol{\beta}$ is rewriteable w.r.t. G' then $\mathfrak{s}p(\boldsymbol{\alpha}, \boldsymbol{\beta})$ will eventually $\mathfrak{s}$-reduce to a syzygy, hence need not be reduced explicitly and can be discarded.

Remark 3.4. Definition 3.10 is very similar to the way in which the original F5 Algorithm chooses the $\mathfrak{s}$-pairs to actually reduce. The idea behind it is that if $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are elements added to G by algorithm 2 and if $\boldsymbol{\beta}$ has higher signature than $\boldsymbol{\alpha}$ then it has been added later to G than $\boldsymbol{\alpha}$ by algorithm 2. We then expect $\boldsymbol{\beta}$ to be "further regular $\mathfrak{s}$-reduced" than $\boldsymbol{\alpha}$. Thus one expects $\mathfrak{s}$-pairs coming from $\boldsymbol{\beta}$ to be easier to regular $\mathfrak{s}$-reduce than $\mathfrak{s}$-pairs coming from $\boldsymbol{\alpha}$. All currently known efficient methods of choosing which $\mathfrak{s}$-pairs to reduce are summed up in the more general concept of a *rewrite order*, see [18, definition 7.1].

Algorithm 3 F5

Input: An ordered sequence $(f_1, \ldots, f_r)$ in R
Output: A signature Gröbner basis G of $(f_1, \ldots, f_r)$

1: $G \leftarrow \emptyset$
2: $H \leftarrow \emptyset$
3: $P \leftarrow \{\mathbf{e}_1, \ldots, \mathbf{e}_r\}$
4: **while** $P \neq \emptyset$ **do**
5: $\quad a\boldsymbol{\alpha} - b\boldsymbol{\beta} \leftarrow$ the $\leq_{\mathrm{POT}}$-minimal element in P
6: $\quad P \leftarrow P \setminus \{a\boldsymbol{\alpha} - b\boldsymbol{\beta}\}$
7: $\quad$ **if** $a\boldsymbol{\alpha}$ and $b\boldsymbol{\beta}$ are not rewritable w.r.t. G **then**
8: $\quad\quad \boldsymbol{\gamma} \leftarrow$ the result of regular $\mathfrak{s}$-reducing $a\boldsymbol{\alpha} - b\boldsymbol{\beta}$ by G
9: $\quad\quad$ **if** $\boldsymbol{\gamma}$ is not a syzygy **then**
10: $\quad\quad\quad G \leftarrow G \cup \{\boldsymbol{\gamma}\}$
11: $\quad\quad\quad P \leftarrow P \cup \{\mathfrak{s}p(\boldsymbol{\gamma}, \boldsymbol{\beta}) \mid \boldsymbol{\beta} \in G\}$
12: $\quad\quad$ **else**
13: $\quad\quad\quad H \leftarrow H \cup \{\boldsymbol{\gamma}\}$
14: $\quad\quad$ **end if**
15: $\quad$ **end if**
16: **end while**
17: **return** G

We obtain the F5 Algorithm, see [18, theorem 7.1] for correctness.

Remark 3.5. We also track the computed set of syzygies in line 13 of Algorithm 3 since we need this set for the rewrite check in line 7. This rewrite check will also take the Koszul syzygies between elements $\boldsymbol{\alpha}, \boldsymbol{\beta} \in G$ into account. By the definition of $\leq_{\mathrm{POT}}$, to check for rewriteability of an element $a\boldsymbol{\alpha}$ against the Kostul syzygies, we simply need to check whether any element in $\left\{\mathrm{lt}(\overline{\boldsymbol{\delta}})e_{\mathrm{ind}(\boldsymbol{\alpha})} \;\middle|\; \boldsymbol{\delta} \in G, \mathrm{ind}(\boldsymbol{\delta}) < \mathrm{ind}(\boldsymbol{\alpha})\right\}$ divides the signature of $\mathfrak{s}(a\boldsymbol{\alpha})$. If that is the case $a\boldsymbol{\alpha}$ is rewritable by the syzygy criterion.

Remark 3.6. In practice, we combine Algorithm 3 with an *F4-like reduction strategy* (see [22] for an introduction to the F4 Algorithm or [18, section 13] for how to combine this with F5). In essence, this means that in line 5 and 6 we select several $\mathfrak{s}$-pairs at once, organize them and their reducers in a matrix whose rows are these polynomials written in the coordinates of the set of all monomials occuring in the chosen polynomials, and then compute the row echelon form of this matrix. Any non-zero row whose leading term has changed is consequently added to the basis G.

We can now give the key lemma to relate Algorithm 3 to Algorithm 1, namely the following reformulation of corollay 7.1 in [18]. We first introduce the following notation: For $1 \leq k \leq r$ denote by $p_k : R^r \to R$ the projection map

$$\boldsymbol{\alpha} = \sum_{i=1}^{r} \alpha_i \mathbf{e}_i \mapsto p_k(\boldsymbol{\alpha}) := \alpha_k.$$

Recall also that an element $p \in R$ is said to be *top-reducible* by a finite set $S \subset R$ if there exists some $q \in S$ such that $\mathrm{lm}(q) \mid \mathrm{lt}(p)$.

Lemma 3.11. *Suppose that Algorithm 3 is done with considering $\mathfrak{s}$-pairs in index k, i.e. assume Algorithm 3 in an intermediate state where only $\mathfrak{s}$-pairs in index $\geq k+1$ are left. Suppose also that $f_k \notin \langle f_1, \ldots, f_{k-1}\rangle$.*

Let H be the current set of syzygies stored in this intermediate state of Algorithm 3. Define

$$\begin{aligned} G_{k-1} &:= \{\boldsymbol{\alpha} \in G \mid \operatorname{ind}(\boldsymbol{\alpha}) \leq k-1\}; \\ B_k &:= \{p_k(\boldsymbol{\alpha}) \mid \boldsymbol{\alpha} \in H,\ \operatorname{ind}(\boldsymbol{\alpha}) = k\}; \\ A_k &:= \overline{G_{k-1}} \cup B_k. \end{aligned}$$

Then A_k is a Gröbner basis of $(f_1, \ldots, f_{k-1} : f_k)$. Furthermore, no $p \in B_k$ is top-reducible by any element in $\overline{G_{k-1}}$.

Proof. It is clear by [18, Theorem 7.1] that A_k is a Gröbner basis of $(f_1, \ldots, f_{k-1} : f_k)$. To prove that no $p \in B_k$ is top-reducible by any element in $\overline{G_{k-1}}$ note that any such element comes from a syzygy $\boldsymbol{\gamma}$ with $\operatorname{lm}(p)\boldsymbol{e}_k = \mathfrak{s}(\boldsymbol{\gamma})$. This $\boldsymbol{\gamma}$ in turn is the result of fully regular $\mathfrak{s}$-reducing some non-rewritable $a\boldsymbol{\alpha}$ by G. If there would exist some $\boldsymbol{\beta} \in G_{k-1}$ with $\operatorname{lm}(\overline{\boldsymbol{\beta}}) \mid \operatorname{lt}(p)$ then $a\boldsymbol{\alpha}$ would be rewritable using the Koszul syzygy $\overline{\boldsymbol{\beta}}\boldsymbol{e}_k - \overline{\boldsymbol{e}_k}\boldsymbol{\beta}$, a contradiction. This proves that p is not top-reducible by any element in $\overline{G_{k-1}}$. □

Remark 3.7. As mentioned already in Remark 3.1, in practice one replaces each element $\boldsymbol{\alpha} \in R^r$ occuring in Algorithm 3 with $(\mathfrak{s}(\boldsymbol{\alpha}), \overline{\boldsymbol{\alpha}})$. This data is sufficient to implement all rewrite checks and regular $\mathfrak{s}$-reductions. However, in order to exploit Lemma 3.11 as described in the next section we replace $\boldsymbol{\alpha}$ with $(p_{\operatorname{ind}(\boldsymbol{\alpha})}(\boldsymbol{\alpha}), \overline{\boldsymbol{\alpha}})$ instead.

4 The nondegenerate part via F5

In this section we modify Algorithm 3 using Lemma 3.11 to obtain a probabilistic version of Algorithm 1. The main ingredient is Algorithm 4 which obtains for a given ideal $I' \subseteq R$ and an element $f \in R$ a Gröbner basis of $(I' : f^\infty) + \langle f \rangle$ and randomly generated elements in $\sqrt{(I' : (I' : f^\infty))} \setminus I'$. Using Proposition 4.4 these elements will generically suffice to implement lines 6-8 of Algorithm 1.

The basic idea of this algorithm is as follows: Given a Gröbner basis $G' := \{q_1, \ldots, q_m\}$ of I' we run F5 on the sequence $(q_1, \ldots, q_m, f)$. Suppose we then find, via Lemma 3.11, an element $g \notin I'$ with $gf \in I'$. We then compute a signature Gröbner basis for the sequence $(q_1, \ldots, q_m, g)$ before running F5 on the updated sequence $(q_1, \ldots, q_m, g, f)$, using the computed signature Gröbner basis of $(q_1, \ldots, q_m, g)$. However, when calling F5 on the sequence $(q_1, \ldots, q_m, g, f)$, we have thrown out a set of elements S in the index corresponding to f that were computed while running F5 on $(q_1, \ldots, q_m, f)$. These elements are fully reduced with respect to $\{q_1, \ldots, q_m\}$. We would therefore like to replace elements in the index corresponding to f in the sequence $(q_1, \ldots, q_m, g, f)$ with appropriate multiples of elements in S. This is accomplished with the following definition:

Definition 4.1. *Let $G \subset R^r$ be a signature Gröbner basis up to some signature T and let $\boldsymbol{\delta} \in R^r$ with $\mathfrak{s}(\boldsymbol{\delta}) = T$. If $\boldsymbol{\alpha} \in S$ is the element with maximal signature in S satisfying $\mathfrak{s}(b\boldsymbol{\alpha}) = \mathfrak{s}(\boldsymbol{\delta})$ for some monomial $b \in R$ then we say that $b\boldsymbol{\alpha}$ is the S-rewriter of $\boldsymbol{\delta}$. If no such element exists we call $\boldsymbol{\delta}$ itself the S-rewriter of $\boldsymbol{\delta}$.*

Appropriately shifting the index of the elements in S to the one corresponding to f in the sequence $(q_1, \ldots, q_m, g, f)$ we then replace any element that we want to reduce with its S-rewriter in Algorithm 4.

Note that in the pseudocode of Algorithm 4, the module representation is given by the sequence $(q_1, \ldots, q_m, g_1, \ldots, g_r, f)$, i.e. by the morphism

$$R^{m+r+1} \to R$$

$$\boldsymbol{e}_k \mapsto \begin{cases} q_k & \text{if } k \leq m \\ g_k & \text{if } m+1 \leq k \leq m+r \\ f & \text{if } k = m+r+1 \end{cases}$$

One should think of Algorithm 4 being initially called with $Q = S = \emptyset$. The two input arguments Q and S are then only used in the recursive calls of Algorithm 4.

Algorithm 4 satF5

Input: A Gröbner basis $G' = \{q_1, \ldots, q_m\} \subset R$ of some ideal I', a set $Q = \{g_1, \ldots, g_r\} \subset (I' : f^\infty)$, a set $S \subset R^{m+r+1}$, an element $f \in R$

Output: A Gröbner basis G of $(I' : f^\infty) + \langle f \rangle$, random elements $h_1, \ldots, h_r \in \sqrt{(I' : (I' : f^\infty))} \setminus I'$

1: $G \leftarrow \{\mathbf{e}_1, \ldots, \mathbf{e}_m\}, H \leftarrow \emptyset, K \leftarrow \emptyset$
2: $P \leftarrow \{\mathbf{e}_{m+1}, \ldots, \mathbf{e}_{m+r+1}\}$
3: **while** $P \neq \emptyset$ **do**
4: $\quad a\boldsymbol{\alpha} - b\boldsymbol{\beta} \leftarrow$ the $\leq_{\mathrm{POT}}$-minimal element in P
5: $\quad k \leftarrow \mathrm{ind}(a\boldsymbol{\alpha} - b\boldsymbol{\beta})$
6: $\quad P \leftarrow P \setminus \{a\boldsymbol{\alpha} - b\boldsymbol{\beta}\}$
7: $\quad$ **if** $a\boldsymbol{\alpha}$ and $b\boldsymbol{\beta}$ are not rewritable **then**
8: $\quad\quad \boldsymbol{\gamma} \leftarrow$ result of regular $\mathfrak{s}$-reducing S-rewriter$(a\alpha - b\beta, S)$
9: $\quad\quad$ **if** $\boldsymbol{\gamma}$ is not a syzygy **then**
10: $\quad\quad\quad G \leftarrow G \cup \{\boldsymbol{\gamma}\}$
11: $\quad\quad\quad P \cup \{\mathfrak{s}p(\boldsymbol{\gamma}, \boldsymbol{\beta}) \mid \boldsymbol{\beta} \in G\}$
12: $\quad\quad$ **else if** $k \neq m + r + 1$ **then**
13: $\quad\quad\quad H \leftarrow H \cup \{\boldsymbol{\gamma}\}$
14: $\quad\quad$ **else**
15: $\quad\quad\quad g \leftarrow p_i(\boldsymbol{\gamma})$
16: $\quad\quad\quad S \leftarrow S \cup \{\boldsymbol{\gamma} \in G \mid \mathrm{ind}(\boldsymbol{\gamma}) = m + r + 1\}$
17: $\quad\quad\quad G'', K' \leftarrow \mathrm{satF5}\left(\overline{\{\boldsymbol{\gamma} \in G \mid \mathrm{ind}(\boldsymbol{\gamma}) \leq m + r\}}, \{g\}, S, f\right)$
18: $\quad\quad\quad$ **return** $G'', K \cup K'$
19: $\quad\quad$ **end if**
20: $\quad$ **end if**
21: $\quad$ **if** $\min\{\mathrm{ind}(\boldsymbol{\delta}) \mid \boldsymbol{\delta} \in P\} = k + 1$ and $k \neq m + r + 1$ **then**
22: $\quad\quad K \cup \{$a random linear combination of the elements in
23: $\quad\quad\quad\quad \{p_k(\boldsymbol{\gamma}) \mid \boldsymbol{\gamma} \in H;\ \mathrm{ind}(\boldsymbol{\gamma}) = k\}\}$
24: $\quad$ **end if**
25: **end while**
26: **return** $\overline{G}, K$

The algorithm is recursive in nature. Two further differences compared to Algorithm 3 stand out:

1. In line 1, we initialize G to $\{\mathbf{e}_1, \ldots, \mathbf{e}_m\}$, the unit vectors corresponding to $\{p_1, \ldots, p_m\}$, which is a Gröbner basis. Since these elements have lower index than any $\mathfrak{s}$-pair treated in the algorithm any reduction by them of a handled $\mathfrak{s}$-pair is automatically a regular $\mathfrak{s}$-reduction. Thus the correctness is not affected even though $\{\mathbf{e}_1, \ldots, \mathbf{e}_m\}$ is not necessarily a signature Gröbner basis.

2. Note that in the notation of Algorithm 4 an index $k \neq m + r + 1$ corresponds to one of the g_i while the index $m + r + 1$ corresponds to f. We now handle syzygies differently compared to Algorithm 3 based on these two cases:

   (a) In line 15 we have identified an element $g \in (I' : f^\infty)$. Using g, the recursive call in line 17 computes a Gröbner basis for $I'' := I' + \langle g_1, \ldots, g_r, g \rangle$ using the computed Gröbner basis of $I' + \langle g_1, \ldots, g_r \rangle$ before attempting to find more elements $g' \in (I'' : f^\infty) = (I' : f^\infty)$.

(b) Conversely, in line 13 we have identified an element $h \in I'_i := ((I' + \langle g_1, \ldots, g_{i-1}\rangle) : g_i)$ for $i = k-m$. If the if-condition in line 21 is met we know that the set $G' \cup \{p_k(\boldsymbol{\gamma}) \mid \boldsymbol{\gamma} \in H;\ \mathrm{ind}(\boldsymbol{\gamma}) = k\}$ generates I'_i by Lemma 3.11. Then, in line 22, we add a random linear combination of the identified elements in $I'_i \setminus I'$ to K. By the following lemma such an element lies in the radical of $(I' : (I' : f^\infty))$:

Lemma 4.2. *Let $I, J \subseteq R$ be two ideals and let $J = \langle g_1, \ldots, g_t\rangle$. Then*

$$(I : J) \overset{\mathrm{rad}}{=} (I : g_1) \cap ((I + \langle g_1\rangle) : g_2) \cap \cdots \cap ((I + \langle g_1, \ldots, g_{t-1}\rangle) : g_t).$$

Proof. The inclusion "$\subseteq$" is obvious. Not let $p \in R$ be such that

$$p^m \in (I : g_1) \cap ((I + \langle g_1\rangle) : g_2) \cap \cdots \cap ((I + \langle g_1, \ldots, g_{t-1}\rangle) : g_t)$$

for some $m \in \mathbb{N}$. Then we have in particular $p^m g_1 \in I$. Now let $i > 1$. By induction, if for some $k \in \mathbb{N}$ we have $p^k g_j \in I$ for all $j \leq i$ then

$$p^{km} g_{i+1} = p^k f + p^k a_1 g_1 + \cdots + p^k a_i g_i \in I$$

for a suitable $f \in I$, $a_1, \ldots, a_i \in R$ and so $p^{km} \in (I : g_{i+1})$. □

Theorem 4.3. *Algorithm 4 is correct and terminates.*

Proof. The correctness of the output immediately follows from Lemma 3.11 and Lemma 4.2. The fact that we can replace elements by their S-rewriter without affecting correctness follows from Lemma 3.9. The termination follows from the termination of F5 and the fact that $(I' : f^\infty) = (I' : f^t)$ for some sufficiently large $t \in \mathbb{N}$. Hence, at some point, Algorithm 4 does not call line 17 anymore. Once this case is reached Algorithm 4 terminates because Algorithm 3 terminates. □

To finally obtain our F5-based implementation of Algorithm 1 we just replace lines 4 and 5 in Algorithm 1 by the corresponding call to Algorithm 4. This yields Algorithm 5. The set $\mathcal{K}$ in Algorithm 1 is replaced by the union of all sets K obtained by each call to Algorithm 4. This is justified by the following proposition:

Proposition 4.4. *Let $I, J \subseteq R$ be two ideals with $J = \langle g_1, \ldots, g_t\rangle$.*

1. *There exists a Zarisiki-open subset $D \subset \mathbb{K}^t$ such that for any $(a_1, \ldots, a_t) \in D$ we have $(I : J^\infty) = (I : (\sum_{j=1}^t a_j g_j)^\infty)$.*
2. *If $K \overset{\mathrm{rad}}{=} J$ then $(I : K^\infty) \overset{\mathrm{rad}}{=} (I : J^\infty)$.*

Proof. (1) easily follows e.g. from [20, Exercise 15.41]. For (2), if $p \in R$ such that $p^k J^l \in I$ for $k, l \in \mathbb{N}$ then for a suitably large $m \in \mathbb{N}$ we have $K^m \subseteq J^l$ so $p^k K^m \in I$ and hence $p \in \sqrt{(I : K^\infty)}$. □

Theorem 4.5. *Algorithm 5 terminates and is correct, i.e. returns a Gröbner basis of the nondegenerate part of $f_1, \ldots, f_c$ if $c \leq n$.*

Proof. The termination follows from the termination of Algorithm 4. Define $I_j := \langle f_1, \ldots, f_{j-1}\rangle$ and $K_j := (I_{j-1} : (I_{j-1} : f_j^\infty))$. By Proposition 4.4, if the elements in the sets K_j are generic enough, then for the ideal J generated by G after the calls in lines 4 and 5 terminate we have

$$(J : \prod_{j=1}^{k} K_j^\infty) \overset{\mathrm{rad}}{=} (J : \prod_{h \in \mathcal{K}} h^\infty),$$

and the correctness of Algorithm 5 follows from the one of Algorithm 1. □

Algorithm 5 Computation of the nondegenerate part via F5

Input: An ordered sequence $(f_1, \ldots, f_c)$ in R with $c \leq n$
Output: A GB G of the nondegenerate part of $f_1, \ldots, f_c$
1: $G \leftarrow \langle f_1 \rangle$
2: $\mathcal{K} \leftarrow \emptyset$
3: **for** $k \in \{2, \ldots, c\}$ **do**
4: $G, K \leftarrow$ result of calling algorithm 4 on $G, \emptyset, \emptyset$ and f_k
5: $\mathcal{K} \leftarrow \mathcal{K} \cup K$
6: **for** $h \in \mathcal{K}$ **do**
7: $G \leftarrow$ a Gröbner basis of the ideal generated by $\texttt{sat}(G, h)$
8: **end for**
9: **end for**
10: **return** G

5 Implementation and Experiments

5.1 Optimizing our Algorithms

We start by describing some further optimizations to Algorithms 4 and 5. The first optimization is again tied to the Singular Critetion (Lemma 3.9). We left these out of the pseudocode presented in section 4 to make the mathematical presentation clearer.

Further Use of the Singular Criterion We can again use the notion of S-rewriteability to implement a `sat`-routine suited to our needs. To do that, we make a slight modification to Algorithm 4 as follows. Suppose we input a Gröbner basis for an ideal J, $Q = \emptyset$, $S = \emptyset$ and a polynomial h in Algorithm 4. Now, due to the definition of $\leq_{\mathrm{POT}}$, if G is a signature Gröbner basis, then $\{\boldsymbol{\alpha} \in G \mid \mathrm{ind}(\boldsymbol{\alpha}) \leq k\}$ is also a signature Gröbner basis for any $k \in \mathbb{N}$. In particular, we may just return, in the notation of Algorithm 4, $\{\boldsymbol{\alpha} \in G \mid \mathrm{ind}(\boldsymbol{\alpha}) \leq m\}$ which is a Gröbner basis of $(J : h^\infty)$. Let $K \subset R$ be another ideal. Having previously computed a Gröbner basis of $(J : h^\infty)$ we must now compute a Gröbner basis of $(J + K : h^\infty)$. To do that we can use the Singular Criterion as above: The set of elements S in index $m+1$ which are obtained during the computation of $(J : h^\infty)$ via Algorithm 4 can again replace elements in suitable signatures when we want to compute a Gröbner basis for $(J + K : h^\infty)$, using S-rewriteability similarly to how it is used in Algorithm 4.

Probabilistic Optimizations Recall by Remark 3.6 that in practice we select and reduce several $\mathfrak{s}$-pairs at once in line 8 of Algorithm 4. In this case we also find several $g'_1, \ldots, g'_t$ in line 15. Replacing g'_1 by a random linear combination of $g'_1, \ldots, g'_t$ we can now, using Proposition 4.4, perform the following probabilistic optimizations:

1. If, in the recursive call of Algorithm 4 in line 17, we find a syzygy signature $m\mathbf{e}_k$ in the index corresponding to g'_1, we also consider the signatures $m\mathbf{e}_{k+1}, \ldots, m\mathbf{e}_{k+t-1}$ as syzygy signatures during the rewrite checks. This is justified because we generically expect that, for some

   $p \in R$, $pg'_1 \in I'$ if and only if $pg'_i \in I'$ for all $i = 2, \ldots, t$.

2. By the same reason it is justified to only add an element to K in line 22 if the index k corresponds to the index of g'_1. This turns out to reduce the number of additional saturation steps we have to perform in lines 6-8 of Algorithm 5 by a significant amount.

With these optimizations in mind, we implemented both Algorithm 3 and 5 in the programming language Julia [4] with an interface to the Singular.jl Julia-library [16]. An interface to the new computer algebra system OSCAR [39] is planned for the future. The implementation is available at

`https://github.com/RafaelDavidMohr/SignatureGB.jl`

In this implementation we use our own data structures for polynomials and polynomial arithmetic. The linear algebra routines for implementing Algorithms 3 and 4 closely follow the corresponding routines presented in [38]. Additionally, our implementation makes use of the modifications to Algorithm 3 presented in [19]. Currently the implementation works only for fields of finite characteristic.

While our implementation is currently not competitive with optimized implementations of Gröbner basis algorithms such as in Maple [37] or msolve [3], we do make use of some standard optimization techniques in Gröbner basis algorithm implementations such as monomial hash tables and divisor bitmasks (see e.g. [40] for a description of these techniques).

5.2 Experimental Results

We used the following examples to benchmark implementations:

1. Cyclic(8), coming from the classical Cyclic(n) benchmark.
2. Pseudo(n), encoding pseudo-singularities as follows

   $$f_1 = \cdots = f_{n-1} = g_1 \cdots = g_{n-1}$$

   with $f_i \in \mathbb{K}[x_1, \ldots, x_{n-2}, z_1, z_2]$, $f_i \in \mathbb{K}[y_1, \ldots, y_{n-2}, z_1, z_2]$, f_i being chosen as a random dense quadric and g_i equalling f_i when substituting $y_1, \ldots, y_{n-2}$ by $x_1, \ldots, x_{n-2}$.
3. Sos(s, n), encoding the critical points of the restriction of the projection on the first coordinate to a hypersurface which is a sum of s random dense quadrics in $\mathbb{K}[x_1, \ldots, x_n]$.

   $$f, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}, \quad f = \sum_{i=1}^{s} g_i^2.$$

4. Sing(n), encoding the critical points of the restriction of the projection on the first coordinate to a (generically singular) hypersurface which is defined by the resultant of two random dense quadrics A, B in $\mathbb{K}[x_1, \ldots, x_{n+1}]$:

   $$f, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}, \quad f = \text{resultant}(A, B, x_{n+1}).$$

5. The Steiner polynomial system, coming from [6].

All these systems are generated by a number of polynomials equal to the number of variables of the underlying polynomial ring. They all have components of different dimensions, one of those being zero-dimensional, i.e. they have a nontrivial nondegenerate part.

In Table 1 we compare Algorithm 5 and a straightforward implementation of ours of Algorithm 1 in Maple. In this implementation, we saturated an ideal J by an ideal K by picking a random linear combination p of generators of K and saturating J by p using Maple's internal saturation routine. Table 1 shows the improvement of Algorithm 5 over Algorithm 1: While Maple's Gröbner basis engine beats our implementation of Algorithm 3 by a wide margin the ratio between the timings of our F5 implementation and our implementation of Algorithm 5 is much better than the ratio between the time it took to compute a

Gröbner basis in Maple and our Maple implementation of Algorithm 1. To additionally show the overhead of Algorithm 5 over Algorithm 3 we noted the number of arithmetic operations in $\mathbb{K}$ when running each of the two algorithms on the polynomial system in question. Our implementation of Algorithm 5 never takes more than 10 times the number of arithmetic operations Algorithm 3 takes, on certain examples we compare very favorably in terms of arithmetic operations to Algorithm 3.

In Table 2 we compare Algorithm 5 to other decomposition methods available in the computer algebra systems Singular, Maple and Macaulay2 [28]. In Singular there is an elimination method [15] and an implementation of the algorithm for equidimensional decomposition presented in [21]. In Maple we compared against the Regular Chains package [9; 10]. In Macaulay2 one is able to compute the intersection of all components of non-minimal dimension again with the method presented in [21]. We then saturated the original ideal by the result to obtain the nondegenerate part. On a high level, our algorithm works similarly, incrementally obtaining information about the component of higher dimension and then removing it via saturation. One should keep in mind that all of these methods, compared to Algorithm 5, work more generally: Except for what we tried in Macaulay2 they are all able to obtain a full equidimensional decomposition of the input ideal.

We gave all of these methods at least an hour for each polynomial system and at most roughly 50 times the time our implementation of Algorithm 5 took. We indicated when these times were exceeded by using ">" in Table 2. We computed all examples on a single Intel Xeon Gold 6244 CPU @ 3.60GHz with a limit of 200G memory. If this limit was exceeded, or if another segfault occured, we indicate it with 'segfault' in Table 2.

Table 1: Comparing Algorithm 1 and Algorithm 5

	Alg. 3 arit. op.	Alg. 5 arit. op.	Alg. 3	Alg. 5	GB in Maple	Alg. 1 in Maple
Cyclic 8	$12*10^9$	$13*10^{10}$	4m	40m	1.2s	154m
Pseudo(2, 12)	$53*10^6$	$31*10^7$	1.16s	5.2s	0.268s	3.44s
Sing(2, 10)	$56*10^6$	$65*10^6$	1.9s	2.9s	0.11s	1.642s
Sing(2, 9)	$25*10^6$	$29*10^6$	1.1s	1.4s	0.06s	0.788s
Sos(2,5,4)	$13*10^7$	$11*10^7$	8.5s	7.3s	0.022s	0.479s
Sos(2,6,3)	$21*10^6$	$21*10^6$	1.11s	1.4s	0.021s	0.261s
Sos(2,6,4)	$48*10^8$	$38*10^8$	148s	169s	0.172s	22.7s
Sos(2,6,5)	$42*10^8$	$20*10^8$	75s	43s	0.458s	10.38s
Sos(2,7,3)	$13*10^7$	$67*10^7$	5.2s	41s	0.047s	7.162s
Sos(2,7,4)	$65*10^8$	$45*10^9$	3m	32m	0.433s	1h
Sos(2,7,5)	$72*10^9$	$35*10^{10}$	25m	20h	2.294s	>359h
Sos(2,7,6)	$17*10^{11}$	$30*10^{11}$	31h	73h	14.348s	5.5h
Steiner	$31*10^9$	$23*10^{10}$	4.2m	42m	27s	13m

Table 2: Comparing with other Decomposition Methods

	Algorithm 5	Singular: Elimination Method	Singular: Algorithm in [21]	Maple: Regular Chains	Macaulay2
Cyclic 8	40m	segfault	>35h	>35h	>35h
Pseudo(2, 10)	0.3s	40s	>1h	>1h	>1h
Pseudo(2, 12)	5.2s	>1h	>1h	>1h	>1h
Pseudo(2, 6)	0.008s	<1s	<1s	0.29s	0.07s
Pseudo(2, 8)	0.03s	<1s	23m	5.82s	13.78s
Sing(2, 10)	2.9s	>1h	>1h	>1h	>1h
Sing(2, 4)	0.02s	1s	>1h	91.32s	0.42s
Sing(2, 5)	0.07s	4s	>1h	>1h	1.94s
Sing(2, 6)	0.15s	56s	>1h	>1h	16.64s
Sing(2, 7)	0.35s	8m	>1h	>1h	289s
Sing(2, 8)	0.68s	23m	>1h	>1h	>1h
Sing(2, 9)	1.4s	>1h	>1h	>1h	>1h
Sos(2,4,2)	0.03s	<1s	<1s	19.4s	0.16s
Sos(2,4,3)	0.03s	1s	3m	14m	0.63s
Sos(2,5,2)	0.02s	<1s	>1h	>1h	0.37s
Sos(2,5,3)	0.34s	>1h	>1h	>1h	9.35s
Sos(2,5,4)	7.3s	>1h	>1h	>1h	183s
Sos(2,6,2)	0.17s	<1s	>1h	>1h	0.7s
Sos(2,6,3)	1.4s	>1h	>1h	>1h	107s
Sos(2,6,4)	169s	>140m	>140m	>140m	>140m
Sos(2,6,5)	43s	>1h	>1h	>1h	>1h
Sos(2,7,2)	2.91s	<1s	>1h	2.94s	0.18s
Sos(2,7,3)	41s	>1h	>1h	>1h	>1h
Sos(2,7,4)	32m	>26h	segfault	>26h	>26h
Sos(2,7,5)	20h	segfault	segfault	>200h	>200h
Sos(2,7,6)	73h	segfault	segfault	>334h	>500h
Steiner	42m	>50h	segfault	>50h	>50h

References

[1] P. Aubry, D. Lazard, and M. Moreno Maza. "On the Theories of Triangular Sets". In: *J. Symb. Comput.* 28.1 (1999), pp. 105–124.

[2] T. Becker and V. Weispfenning. *Gröbner bases*. Vol. 141. Graduate Texts in Mathematics. A computational approach to commutative algebra, In cooperation with Heinz Kredel. Springer-Verlag, New York, 1993.

[3] J. Berthomieu, C. Eder, and M. Safey El Din. "msolve: A Library for Solving Polynomial Systems". In: *ISSAC'21*. Saint Petersburg, Russia, 2021.

[4] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. "Julia: A fresh approach to numerical computing". In: *SIAM review* 59.1 (2017), pp. 65–98.

[5] W. Bosma, J. Cannon, and C. Playoust. "The Magma Algebra System. I. The User Language". In: *J. Symbolic Comput.* 24.3-4 (1997), pp. 235–265.

[6] P. Breiding, B. Sturmfels, and S. Timme. "3264 conics in a second". In: *Notices Amer. Math. Soc.* 67.1 (2020), pp. 30–37.

[7] M. Caboara, P. Conti, and C. Traverse. "Yet Another Ideal Decomposition Algorithm". In: *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*. Lecture Notes in Computer Science. Springer, 1997, pp. 39–54.

[8] C. Chen, O. Golubitsky, F. Lemaire, M. M. Maza, and W. Pan. "Comprehensive triangular decomposition". In: *International Workshop on Computer Algebra in Scientific Computing*. Springer. 2007, pp. 73–101.

[9] C. Chen, F. Lemaire, M. M. Maza, W. Pan, and Y. Xie. "Efficient computations of irredundant triangular decompositions with the regularchains library". In: *International Conference on Computational Science*. Springer. 2007, pp. 268–271.

[10] C. Chen and M. Moreno Maza. "Algorithms for computing triangular decomposition of polynomial systems". In: *J. Symbolic Comput.* 47.6 (2012). Advances in Mathematics Mechanization, pp. 610–642.

[11] S.-C. Chou and X.-S. Gao. "Ritt-Wu's Decomposition Algorithm and Geometry Theorem Proving". In: *10th International Conference on Automated Deduction*. Lecture Notes in Computer Science. Springer, 1990, pp. 207–220.

[12] X. Dahan, X. Jin, M. M. Maza, and É. Schost. "Change of order for regular chains in positive dimension". In: *Theoretical Computer Science* 392.1-3 (2008), pp. 37–65.

[13] X. Dahan, M. Moreno Maza, E. Schost, W. Wu, and Y. Xie. "Lifting techniques for triangular decompositions". In: *ISSAC'05*. ACM, New York, 2005, pp. 108–115.

[14] W. Decker, G.-M. Greuel, and G. Pfister. "Primary Decomposition: Algorithms and Comparisons". In: *Algorithmic Algebra and Number Theory*. Springer, 1999, pp. 187–220.

[15] W. Decker, G.-M. Greuel, and G. Pfister. "Primary Decomposition: Algorithms and Comparisons". In: *Algorithmic Algebra and Number Theory*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 187–220.

[16] W. Decker, G.-M. Greuel, G. Pfister, and H. Schönemann. *Singular 4-3-0 — A Computer Algebra System for Polynomial Computations*. 2022.

[17] J. Della Dora, C. Dicrescenzo, and D. Duval. "About a New Method for Computing in Algebraic Number Fields". In: *Research Contributions from the European Conference on Computer Algebra-Volume 2*. EUROCAL '85. Springer-Verlag, 1985, pp. 289–290.

[18] C. Eder and J.-C. Faugère. "A survey on signature-based algorithms for computing Gröbner bases". In: *J. Symbolic Comput.* 80 (2017), pp. 719–784.

[19] C. Eder and J. Perry. "F5C: A variant of Faugère's F5 algorithm with reduced Gröbner bases". In: *J. Symbolic Comput.* 45.12 (2010), pp. 1442–1458.

[20] D. Eisenbud. *Commutative Algebra: with a View Toward Algebraic Geometry*. New York, NY: Springer New York, 1995.

[21] D. Eisenbud, C. Huneke, and W. Vasconcelos. "Direct Methods for Primary Decomposition". In: *Invent. Math.* 110.1 (1992), pp. 207–235.

[22] J.-C. Faugère. "A new efficient algorithm for computing Gröbner bases (F4)". In: *Journal of Pure and Applied Algebra* 139.1 (1999), pp. 61–88.

[23] J.-C. Faugère. "A new efficient algorithm for computing Gröbner bases without reduction to zero (F5)". In: *ISSAC'02*. 2002, pp. 75–83.

[24] J.-C. Faugère. "Finding All the Solutions of Cyclic 9 Using Gröbner Basis Techniques". In: *Computer Mathematics*. ASCM 2001. World Scientific, 2001, pp. 1–12.

[25] P. Gianni, B. Trager, and G. Zacharias. "Gröbner Bases and Primary Decomposition of Polynomial Ideals". In: *J. Symb. Comput.* 6.2 (1988), pp. 149–167.

[26] M. Giusti, G. Lecerf, and B. Salvy. "A Gröbner Free Alternative for Polynomial System Solving". In: *J. Complexity* 17.1 (2001), pp. 154–211.

[27] H.-G. Gräbe. "Minimal primary decomposition and factorized Gröbner bases". In: *Appl. Algebra Engrg. Comm. Comput.* 8.4 (1997), pp. 265–278.

[28] D. R. Grayson and M. E. Stillman. *Macaulay2, a Software System for Research in Algebraic Geometry*. Available at http://www.math.uiuc.edu/Macaulay2/.

[29] G.-M. Greuel and G. Pfister. *A Singular Introduction to Commutative Algebra*. 2nd ed. Springer Berlin Heidelberg, 2007.

[30] E. Hubert. "Notes on Triangular Sets and Triangulation-Decomposition Algorithms I". In: *Symbolic and Numerical Scientific Computation*. Lecture Notes in Computer Science. Springer, 2003, pp. 1–39.

[31] G. Jeronimo and J. Sabia. "Effective Equidimensional Decomposition of Affine Varieties". In: *Journal of Pure and Applied Algebra* 169.2 (2002), pp. 229–248.

[32] T. Krick and A. Logar. "An Algorithm for the Computation of the Radical of an Ideal in the Ring of Polynomials". In: *AAECC 1991*. Springer-Verlag, 1991, pp. 195–205.

[33] G. Lecerf. "Computing an equidimensional decomposition of an algebraic variety by means of geometric resolutions". In: *ISSAC'00*. 2000, pp. 209–216.

[34] G. Lecerf. "Computing the Equidimensional Decomposition of an Algebraic Closed Set by Means of Lifting Fibers". In: *Journal of Complexity* 19.4 (2003), pp. 564–596.

[35] F. Lemaire, M. Moreno Maza, W. Pan, and Y. Xie. "When does $<T>$ equal sat(T)?" In: *J. Symb. Comput.* 46.12 (2011), pp. 1291–1305.

[36] X. Li, M. Moreno Maza, and W. Pan. "Computations modulo regular chains". In: *ISSAC'09*. ACM, New York, 2009, pp. 239–246.

[37] Maplesoft, a division of Waterloo Maple Inc.. *Maple*. Version 2021. Waterloo, Ontario, 2021.

[38] M. Monagan and R. Pearce. "A compact parallel implementation of F4". In: *Proceedings of the 2015 International Workshop on Parallel Symbolic Computation*. 2015, pp. 95–100.

[39] *OSCAR – Open Source Computer Algebra Research system, Version 0.7.1*. The OSCAR Team, 2022.

[40] B. H. Roune and M. Stillman. "Practical Gröbner basis computation". In: *ISSAC'12*. 2012, pp. 203–210.

[41] D. Wang. "An Elimination Method for Polynomial Systems". In: *J. Symb. Comput.* 16.2 (1993), pp. 83–114.

[42] D. Wang. *Elimination Methods*. Texts and Monographs in Symbolic Computation. Springer Vienna, 2001.

[43] W.-T. Wu. "Basic Principles of Mechanical Theorem Proving in Elementary Geometries". In: *J. Autom. Reason.* 2.3 (1986), pp. 221–252.