Linear Algebra for Zero-Dimensional Ideals

Conference Paper · July 2019

DOI: 10.1145/3326229.3326279

1 author:

Anna Maria Bigatti
Università degli Studi di Genova
122 PUBLICATIONS **904** CITATIONS

SEE PROFILE

Linear Algebra for Zero-Dimensional Ideals

Anna Maria Bigatti
Dipartimento di Matematica
Genova, Italy
bigatti@dima.unige.it

ABSTRACT

Given a zero-dimensional ideal I in a polynomial ring, many algorithms start by finding univariate polynomials in I, or by computing a lex-Gröbner basis of I. These are related to considering the **minimal polynomial** of an element in P/I, which may be computed using Linear Algebra from a Gröbner basis (for any term-ordering). In this tutorial we'll see algorithms for computing minimal polynomials, applications of modular methods, and then some applications, namely algorithms for computing radicals and primary decompositions of zero-dimensional ideals, and also for testing radicality and maximality.

We'll also address a kind of opposite problem: given a "geometrical description", such as a finite set of points, find the ideal of polynomials which vanish at it. We start from the original **Buchberger-Möller algorithm**, and we show some developments.

All this will be done with a special eye on the practical implementations, and with demostrations in CoCoA.

KEYWORDS

Zero-dimensional ideals, minimal and characteristic polynomial, primary decomposition, radical ideals, solving polynomial systems, ideals of points, border bases, CoCoA, CoCoALib.

ACM Reference Format:
Anna Maria Bigatti. 2019. Linear Algebra for Zero-Dimensional Ideals. In *International Symposium on Symbolic and Algebraic Computation (ISSAC '19), July 15–18, 2019, Beijing, China.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3326229.3326279

1 INTRODUCTION

Let K be a field, and $P = K[x_1, \ldots, x_n]$ the polynomial ring with n indeterminates.

The easiest geometric object in the affine space K^n is a single rational point. It has no secrets, in particular its defining ideal, i.e. the set of all the polynomials which vanish at the point, is straightforward to describe. Namely, for a point with coordinates $(a_1, \ldots, a_n)$, the corresponding ideal is $\langle x_1 - a_1, \ldots, x_n - a_n \rangle \subseteq P$.

The matter becomes more complicated when we consider the defining ideal $I(\mathbb{X})$ of a finite set of points $\mathbb{X}$, maybe not rational, maybe with some *multiplicities*.

ISSAC '19, July 15–18, 2019, Beijing, China

ACM ISBN 978-1-4503-6084-5/19/07…$15.00
https://doi.org/10.1145/3326229.3326279

In this tutorial we explore these two opposite points of view: given a zero-dimensional ideal, what can we say about the points it vanishes on? given a set of rational points $\mathbb{X} = \{P_1, \ldots, P_s\}$, *i.e.* having coordinates in K, how can we compute its defining ideal $I(\mathbb{X})$?

1.1 From algebra to geometry

It is well-known that if R is a zero-dimensional affine K-algebra (*i.e.* a zero-dimensional algebra of the form $R = K[x_1, \ldots, x_n]/I$) then R is a finite-dimensional K-vector space. 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 [19], where the particular importance of **minimal polynomials** emerges quite clearly. The book also suggests several algorithms which use minimal polynomials as a crucial tool. We investigated their theory, their uses, and their applications in Abbott et al. [4].

In this tutorial we'll see how CoCoA computes minimal polynomials using Linear Algebra, and some intriguing aspects in the implementation of the derived algorithms.

1.2 From geometry to algebra

As to the second question, we observe that $I(\mathbb{X}) = \bigcap_{i=1}^{s} I(P_i)$, hence it is possible to use a well-known byproduct of Buchberger's Algorithm to compute the intersection. However, this approach is not efficient from the computational point of view and to remedy this an algorithm (called **Buchberger-Möller algorithm**) of low complexity was presented by Buchberger and Möller [12]; moreover, this algorithm has good performance (see for instance Marinari et al. [21]). Several generalizations have since been proposed for instance to the cases of points with multiplicity (for instance Lakshman [20] and Abbott et al. [7]), of points described by differential conditions, and for identifying *almost-vanishing* polynomials for sets of *approximate points* (Abbott et al. [6] and Fassino [13])

Why are we interested in the defining ideal $I(\mathbb{X})$? We answer by listing just a few examples where ideals of points have been used in different fields of mathematics.

- They encode fundamental properties of algebraic varieties of which they are linear sections (see for instance Geramita et al. [15] and the references indicated there).
- They are the basis for interpolation problems of numerical analysis (see Möller [22]).
- They are used in coding theory (see Sakata [31]).
- They have been recently introduced into the realm of statistics, where finite sets of points are called designs and fractions (see Robbiano [30]).

In this tutorial we'll see how CoCoA computes ideals of points using Linear Algebra, and how it computes with approximate points.

1.3 Modular methods

Notoriously, when using rational coefficients, *i.e.* $K = \mathbb{Q}$, although the input ideal/points have simple coefficients/coordinates, the computations may become very cumbersome. In such cases the technique of using modular reduction has a long tradition, for example in univariate polynomial factorization, and in exact linear algebra.

More specifically, regarding multivariate polynomial ideals, see the articles by Pauer [29], Winkler [33], Gräbe [16], Noro and Yokoyama [25], Noro and Yokoyama [26], Noro and Yokoyama [27], Aoyama and Noro [8], Noro [24], Idrees et al. [17] and Monagan [23]. But in this context, usually, modular methods are associated with results which are correct *only with high probability*. In contrast, we show how to apply them to the computation of the minimal polynomial of a zero-dimensional ideal, and of the ideal of a set of points, and guarantee the correctness of the result.

In this tutorial we use repeatedly the notion of Gröbner basis. Needless to say this is now one of the main ingredients in Computational Algebra. The layout of the theory of Gröbner bases can be found in the original papers by Buchberger ([9], [10], and [11]), and in many books, for example Kreuzer and Robbiano [18].

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 in this first part.

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 $\mu(z) = \sum_{i=0}^{d} \lambda_i z^i \in K[z]$ which vanishes modulo I when evaluated at $z = f$, *i.e.* $\mu(f) \in I$. For example, in the quotient ring $R = \mathbb{Q}[x, y]/(x^2, y^2)$ the minimal polynomial for the residue class of $x + y$ is $\mu(z) = z^3$.

The first step was to implement in CoCoALib algorithms for computing the minimal polynomial of an element of R or that of a K-endomorphism of R. These functions are also accessible from CoCoA-5: type `?MinPolyQuot` for documentation.

Here is an example where we verify that the algebraic extension made by $\{\sqrt{2}, \sqrt[3]{3}, \sqrt[5]{5}, \sqrt[7]{7}\}$ is actually a field, and that $\sqrt{2} + \sqrt[3]{3} + \sqrt[5]{5} + \sqrt[7]{7}$ is indeed a primitive element for the extension:

```
/**/ use P ::= QQ[a,b,c,d];
/**/ I := ideal(a^2-2, b^3-3, c^5-5, d^7-7);
/**/ IsMaximal(I); --> Is P/I is a field?
true
/**/ multiplicity(P/I);  -- dim_QQ(P/I)
210
/**/ mu := MinPolyQuot(a+b+c+d, I, a);
/**/ deg(mu);
210
/**/ mu;
a^210 -210*a^208 -210*a^207 +21840*a^206 +41370*a^205
-1477945*a^204 -4032210*a^203 +72457350*a^202
+257727120*a^201 -2722292727*a^200 -12102798750*a^199
+81019988455*a^198 +444258452820*a^197 ............
```

Remark 2.1. The particular case of $\mu_{x_i}(x_i)$, where x_i is an indeterminate, is a very important and popular object when computing: in fact $\mu_{x_i}(x_i)$ is the lowest degree polynomial in x_i belonging to I, that is $I \cap K[x_i] = \langle \mu_{x_i}(x_i) \rangle$.

Remark 2.2 (Elimination)**.** A well-known method for computing the minimal polynomial $\mu_f(z)$ is by elimination. One extends P with a new indeterminate to produce a new polynomial ring $R = K[x_1, \ldots, x_n, z]$, then defines the ideal $J = IR + \langle z - f \rangle$ in R, and finally eliminates the indeterminates $x_1, \ldots, x_n$.

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 5.1).

Roughly, this is what the very definition suggests:

Algorithm MinPolyQuotDef

$P = K[x_1, \ldots, x_n]$, term-ordering σ

Input: I a zero-dimensional ideal in P, f polynomial in P

- compute GB, the σ-Gröbner basis for I
  from GB compute QB, the monomial quotient basis of P/I
- let $r_0 = f^0$ $(= 1)$
- ***Main Loop:*** for $i = 1, 2, \ldots, \text{len}(QB)$ do
  - compute $r_i = \text{NF}(f^i)$
  - if there is a linear dependency $r_i = \sum_{j=0}^{i-1} c_j r_j$ with coefficients $c_j \in K$ **return** $z^i - \sum_{j=0}^{i-1} c_j z^j$

Output: $\mu_f(z) \in K[z]$

This is how to mimic these steps in CoCoA on a trivial example:

```
/**/ use P ::= QQ[x,y];
/**/ L := ***[y^2-6y+8, xy-2x-5y+10, x^2-4x-4y+11]***;
/**/ I := ideal(L);
/**/ GBasis(I);
[y^2 -6*y +8,  x*y -2*x -5*y +10,  x^2 -4*x -4*y +11]
/**/ QuotientBasis(I); -------->  [1,  y,  x]
/**/ y^0;         --> 1       -->  [1,  0,  0]
/**/ NF(y^1, I); --> y       -->  [0,  1,  0]
/**/ NF(y^2, I); --> 6*y -8 --> [-8,  6,  0]
/**/ MinPolyQuot(y, I, y); -----> is indeed....
y^2 -6*y +8
```

A simple but powerful optimization is given by observing that $\text{NF}(f^i) = \text{NF}(f \cdot r_{i-1})$, which makes good use of previous reduction on r_{i-1} and avoids computing powers with possibly large exponents (up to $\dim_K(P/I)$). Moreover, incrementally keeping in memory the matrix row-reduction, we obtained in CoCoA a satisfactory implementation for $K = \mathbb{F}_p$. See Section 2 of [4] for details.

3 MODULAR METHODS AND RATIONAL RECONSTRUCTION

In this section we consider $P = \mathbb{Q}[x_1, \ldots, x_n]$.

Given a positive $\delta \in \mathbb{N}$, 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 $k \in \mathbb{N}$.

Definition 3.1. Let δ be a positive integer, and p be a prime number not dividing δ. We write $\boldsymbol{\pi_p}$ to denote both the canonical homomorphism $\mathbb{Z}_\delta \longrightarrow \mathbb{F}_p$ and its natural "coefficientwise" extensions to $\mathbb{Z}_\delta[x_1, \ldots, x_n] \longrightarrow \mathbb{F}_p[x_1, \ldots, x_n]$.

Having to reduce modulo p a set of polynomials G, it is quite natural to say that p is a *usable* prime if it does not divide the least common multiple δ of all the denominators in the coefficients in G.

Then, among these *usable* primes, there are some *bad* primes, *i.e.* primes p such that the result over $\mathbb{F}_p$ starting from $\pi_p(G)$ is not the modular image of the result over $\mathbb{Q}$ starting from G.

In a great variety of problems, it is true that all but finitely many primes are then *good*, and that randomly chosen primes are *good* with very high probability. In the literature, several authors have looked at various notions of bad reduction in similar contexts. However, our approach is systematically tied to reduced Gröbner bases as computationally robust representations of ideals; we study more deeply this approach in the forthcoming paper [5].

Definition 3.2. Given a non-zero ideal I in P, we define $\mathbf{den_\sigma(I)}$, the **σ-denominator of I**, to be the least common multiple of all denominators in G_σ, the reduced σ-Gröbner basis of I.

Then, for a prime p not dividing $den_\sigma(I)$, we define $\mathbf{I_{(p,\sigma)}}$, the **(p, σ)-reduction of I**, as the ideal generated by the set $\pi_p(G_\sigma)$, in the polynomial ring $\mathbb{F}_p[x_1, \ldots, x_n]$.

We have that the set $\pi_p(G_\sigma)$ is indeed the reduced σ-Gröbner basis of the ideal $\langle \pi_p(G_\sigma) \rangle$. Thus, if I is zero-dimensional, so is $I_{(p,\sigma)}$.

The key for guaranteeing the result is that, for any *usable* prime p, we proved that $\pi_p(\mu_f(z))$ is a multiple of $\mu_{\pi_p(f)}(z)$. In other words, the degree of the minimal polynomial after the modular computation, is $\leq \deg(\mu_f(z))$.

Therefore, after performing the computation with some *usable* primes, we can recognize as *bad* primes all those whose modular results have degree is smaller than the maximum.

Finally, by using the rational reconstruction described in Abbott [1], we reconstruct the rational minimal polynomial over $\mathbb{Q}$. At this point we verify it actually vanishes on f and, if it does, it is guaranteed to be the correct answer (again by reasoning on the degree).

The combination of the theoretical results explained in section with various implementation details lead to good practical performance. See Section 3 of [4] for details.

4 MANY APPLICATION OF MINIMAL POLYNOMIALS

As mentioned earlier, an efficient way of computing minimal polynomials lets us compute quickly a number of other important invariants of zero-dimensional affine K-algebras. For instance, in [4] we described algorithms for testing whether a zero-dimensional ideal is radical, maximal or primary, and others for computing the radical and the primary decomposition of a zero-dimensional ideal.

All the algorithms described have been implemented in CoCoA. Their detailed description is in Section 4 in [4].

In particular, it is noteworthy that in the case of coefficient fields of (small) finite characteristic, Frobenius spaces play a fundamental role, as described in [19, 28], and provide special algorithms.

Here we consider $P = \mathbb{Q}[x_1, \ldots, x_n]$, and we collect these crucial results from [4]:

Proposition 4.1. *Let $\ell \in P$ the generic linear form.*

(1) *If I is maximal, $\bar{\ell}$ is a primitive element of the field P/I.*
(2) *If I is radical, then* $\deg(\mu_\ell(z)) = \dim_{\mathbb{Q}}(P/I)$.
(3) *If $\mu_f(z)$ is square-free and its degree is* $\dim_K(P/I)$ *then I is a radical ideal.*

4.1 IsMaximal(I): practically effective NON-algorithm!

Algorithm IsMaximal

Input $\boldsymbol{I}$, an ideal in P

- *Loop:* repeat
  - pick a **random** linear form $\ell \in P$; compute μ_ℓ
  - if μ_ℓ is reducible then **return false**
  - if $\deg(\mu_\ell) = d$ then **return true**

Output *true/false* indicating the maximality of $\boldsymbol{I}$.

Remark 4.2. Note that IsMaximal is not an algorithm because termination is not guaranteed. But in practice if ℓ a **random** linear form then it behaves as the **generic** linear form.

In practice, though, we'd better check the minimal polynomials of the indeterminates x_i's before performing the "generic" loop: the computation of μ_{x_i} is generally much faster, and often we get the final answer with one of them.

4.2 CoCoA's Technical utilities: interrupt, verbosity, timeout

Now we go for a little detour into CoCoA technicalities.

Everyone knows that a Gröbner basis computation may be taking a long time, and it is useful to developers (and also to the curious), to understand what is happening inside important functions.

The CoCoA function `SetVerbosityLevel(N)` sets the global verbosity level to `N`. Various functions defined in CoCoALib and in the CoCoA packages print out some internal progress messages when the global verbosity level is higher than a certain threshold value. Higher levels trigger greater verbosity, and the manual entries for each function list the relevant thresholds.

For instance, the lowest level giving information on the progress of Gröbner bases is 100.

In some hard Gröbner basis computations one may see that the number of pairs yet to be processed is unfeasibly high. The user may then choose to interrupt the computation by typing `Ctrl-C`: the computation will be interrupted as soon as the reduction of the current S-polynomial terminates. The user may instead set a *timeout*: the calculation is automatically interrupted after the specified computation time.

This interruption cancels the incomplete Gröbner basis computation, and returns the computer to the state it was in just before the Gröbner basis computation was begun. Doing this correctly is not entirely trivial: thanks to its clean, exception-safe design, CoCoALib guarantees that no memory has been lost, and no data has been modified.

Now we see how to fruitfully use CoCoA's timeout mechanism for the computation of radical ideals.

4.3 0-dim Radical: classic, new, and timeout

We indicate by **rad(μ)**, sometimes called *square-free of* μ, the product of all distinct irreducible factors of μ, without multiplicities), and we compare the *classic* algorithm for computing the radical of a zero-dimensional ideal, with our new approach

Algorithm Radical0Dim – classic

Input I, a zero-dimensional ideal in P
1 – nothing –
2 *Main Loop:* for each indeterminate x_i do
2.1 compute $\mu_i = \mu_{x_i}$
2.2 if μ_i is not square-free then
2.2.1 let $\mu_i = \text{rad}(\mu_i)$
3 **return** $I + \langle \mu_1, \ldots, \mu_n \rangle$
Output the radical of I

Now, using Proposition 4.1, we can stop the computation earlier, if we detect some minimal polynomials with degree equal to $\dim_{\mathbb{Q}}(P/\mathbf{J})$. But it may also happen that a nice easy input ideal I becomes an awful ideal J after adding some $\text{rad}(\mu_i)$. So we combine the proposition with the timeout:

Algorithm Radical0Dim

Input I, a zero-dimensional ideal in P
1 let $\mathbf{J} = I$ and compute $d = \dim_{\mathbb{Q}}(P/\mathbf{J})$
2 *Main Loop:* for each indeterminate x_i do
2.1 compute $\mu = \mu_{x_i}$ (w.r.t J or I!)
2.2 if μ is not square-free then
2.2.1 let $\mu = \text{rad}(\mu)$
2.2.2 let $\mathbf{J} = \mathbf{J} + \langle \mu(x_i) \rangle$
2.2.3 compute $d = \dim_{\mathbb{Q}}(P/\mathbf{J})$ *if GB is fast enough!!*
2.3 if $\deg(\mu) = d$ then **return J**
3 **return J**
Output the radical of I

4.4 Primary Decomposition

The computation of the primary decomposition of a zero-dimensional ideal computes minimal polynomials "behind the scenes":

```
/**/ use P ::= QQ[x,y,z];
/**/ I := ideal(3*y^2*z^2 +3*x*y^2 +1,
                x^4 +y*z^3,  y^4 +y*z^3)
/**/ PD := PrimaryDecomposition0(I);
/**/ len(PD);
9
```

In Algorithm 5.1.5 in Kreuzer and Robbiano [19] the primary decomposition of a zero-dimensional ideal is obtained by computing the minimal polynomial μ_ℓ of a generic linear form ℓ. Then computing its factorization $\mu_\ell(z) = p_1(z)^{m_1} \cdots p_s(z)^{m_s}$, and finally returning the primary components $Q_i = I + \langle p_i^{m_i} \rangle$.

Mathematically speaking, this is very elegant and suggests that, with a *good random linear form* we probably get the primary decomposition in just one step (as for IsMaximal). But here we have an additional problem: how can we guarantee that it is *random enough* to get all the primary ideals?

So we developed a fast primality test, as an optimized combination of IsMaximal and Radical0Dim, to check the primality of the output ideals.

Having done that, we again opted to make use of the speed of the computation of the minimal polynomials of the indeterminates x_i's, and also their simpler coefficients, as for we did for IsMaximal. So we implemented a recursive algorithm, cycling on all the indeterminates first, and checking if the components we obtained are primary, and finally using random linear forms only for splitting the remaining non-primary ideals.

It is always quite interesting, and a also a bit painful, when the mathematical elegance of a single computation works less effectively than a coarser recursive approach.

A further interesting application of the computation of primary decompositions of zero-dimensional ideals, is an effective method for factorizing univariate polynomials over finite field extensions (Palezzato [28], with inspiration from Sun and Wang [32] and Kreuzer and Robbiano [19]). See [3] for a detailed example.

5 BUCHBERGER-MÖLLER ALGORITHM

Now we change point of view. Let $\mathbb{X}$ be a non-empty, finite set of points in K^n, then the set of all polynomials in $K[x_1, \ldots, x_n]$ which vanish at all points in $\mathbb{X}$ is an ideal, $I_{\mathbb{X}}$. One reason this ideal is interesting is because it captures the "ambiguity" present in a polynomial function which has been interpolated from its values at the points of $\mathbb{X}$. How best to compute a set of generators for $I_{\mathbb{X}}$, or a Gröbner basis, knowing just the points $\mathbb{X}$?

As we said in the introduction, if $\mathbb{X}$ contains a single point $(a_1, \ldots, a_n)$ then we can write down immediately a Gröbner basis, namely $\langle x_1 - a_1, \ldots, x_n - a_n \rangle$. If $\mathbb{X}$ contains several points we could intersect the ideals for each single point via Gröbner basis computations, but this is quite inefficient.

A far more efficient method is the Buchberger-Möller algorithm [12]. It is based, as in the Algorithm MinPolyQuotDef, on iteratively constructing a matrix whose rows are the evaluations of power-products, generated in increasing order, in the points in $\mathbb{X}$. Then, looking for linear dependencies, we find the polynomials of the Gröbner basis.

5.1 Further developments

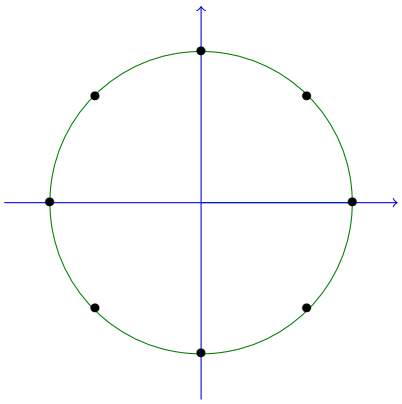
In Abbott et al. [2] there is a detailed complexity analysis of the original algorithm, and also an extension to the projective case.

It was later further generalized to zero-dimensional schemes Abbott et al. [7], where it turned out that it also incorporates the well-known FGLM algorithm.

Remark 5.1. The famous algorithm FGLM, presented in Faugère et al. [14] is again based on iteratively constructing a matrix and looking for linear dependencies. In this case the input is a σ-Gröbner basis of a zero-dimensional ideal I, and we want to compute a Gröbner basis of I with respect to another term-ordering τ.

Much as we have seen in the previous sections for computing with rational coefficients, the Buchberger-Möller algorithm also benefits from a modular approach, and naturally the CoCoA implementation uses this technique.

The use of simple linear algebra in the Buchberger-Möller algorithm makes it a good candidate for identifying "almost-vanishing" polynomials for sets of *approximate points:* for instance, the points in the example above *almost lie on* a circle of radius 9.95 centred on the origin, though we cannot tell this from the *exact* Gröbner basis: $\{x^2y + \frac{49}{51}y^3 - \frac{4900}{51}y, \quad x^3 + \frac{51}{49}xy^2 - 100x,$
$y^4 - \frac{2499}{2}x^2 - \frac{2699}{2}y^2 + 124950, \quad xy^3 - 49xy\}$.



In fact, the notion of Gröbner basis does not generalize well to an approximate context because the algebraic structure of a Gröbner basis is determined by Zariski-closed conditions (*i.e.* the structure is valid when certain polynomials vanish); instead, the notion of a *Border Basis* is better suited since the validity of its structure depends on a Zariski-open condition (*i.e.* provided a certain polynomial *does not* vanish). So long as the approximate points are not too few nor too imprecise the Numerical Buchberger-Möller algorithm can compute at least a partial Border Basis, and this should identify any "approximate polynomial conditions" which the points almost satisfy (see Abbott, Fassino, Torrente [6] and Fassino [13]). We can ask CoCoA to allow a certain approximation on the coordinates of the points: allowing an approximation on both coordinates of $\varepsilon = 0.1$ we find that the points are *almost on a circle*, more precisely, on the conic $x^2 + \frac{4999}{5001}y^2 - \frac{165000}{1667} = 0$. On the other hand, allowing maximum approximation $\varepsilon = 0.01$, the points are not *ε-near* any conic.

```
/**/ epsilon := [0.1, 0.1];
/**/ AP01 := ApproxPointsNBM(P, mat(points), mat([epsilon]));
/**/ indent(AP01.AlmostVanishing);
[
  x^2 +(4999/5001)*y^2 -165000/1667, // almost a circle
  x*y^3 -49*x*y,
  y^5 -149*y^3 +4900*y
]

/**/ epsilon := [0.01, 0.01]; // coord approximation 0.01
/**/ AP001 := ApproxPointsNBM(P, mat(points), mat([epsilon]));
/**/ indent(AP001.AlmostVanishing);
[
  x^2*y +(49/51)*y^3 +(-4900/51)*y,
  x^3 +(51/49)*x*y^2 -100*x,
  y^4 +(-2499/2)*x^2 +(-2699/2)*y^2 +124950,
  x*y^3 -49*x*y
]
```

REFERENCES

[1] John Abbott. 2017. Fault-Tolerant Modular Reconstruction of Rational Numbers. *J. Symb. Comp.* 80 (2017), 707–718.

[2] John Abbott, Anna M. Bigatti, Martin Kreuzer, and Lorenzo Robbiano. 2000. Computing Ideals of Points. *J. Symb. Comput.* 30, 4 (2000), 341–356.

[3] John Abbott, Anna Maria Bigatti, and Elisa Palezzato. 2018. 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. 88–94.

[4] John Abbott, Anna Maria Bigatti, Elisa Palezzato, and Lorenzo Robbiano. 2019. Computing and Using Minimal Polynomials. *J. Symb. Comput.* To appear (2019).

[5] John Abbott, Anna Maria Bigatti, and Lorenzo Robbiano. 2017. Implicitization of hypersurfaces. *J. Symb. Comput.* 81 (2017), 20–40.

[6] John Abbott, Claudia Fassino, and Maria Laura Torrente. 2008. Stable Border Bases for Ideals of Points. *J. Symb. Comput.* 43, 12 (2008), 375–392.

[7] John Abbott, Martin Kreuzer, and Lorenzo Robbiano. 2005. Computing zero-dimensional schemes. *J. Symb. Comput.* 39, 1 (2005), 31–49.

[8] Toru Aoyama and Masayuki Noro. 2018. Modular Algorithms for Computing Minimal Associated Primes and Radicals of Polynomial Ideals. In *Proc. ISSAC 2018*. 31–38.

[9] Bruno Buchberger. 1965. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal.* Ph.D. Dissertation.

[10] Bruno Buchberger. 1970. Ein algorithmisches Kriterium für die Lösbarkeit eines algebraischen Gleichungssystems. *Aequationes mathematicae* 4, 3 (1970), 374–383.

[11] Bruno Buchberger. 1985. Gröbner-Bases: An Algorithmic Method in Polynomial Ideal Theory. , 184–232 pages.

[12] Bruno Buchberger and Hans Michael Möller. 1982. The Construction of Multivariate Polynomials with Preassigned Zeros, In EUROCAM '82 Proceedings of the European Computer Algebra Conference on Computer Algebra. *Lecture Notes in Computer Science* 144, 24–31.

[13] Claudia Fassino. 2010. Almost Vanishing Polynomials for Sets of Limited Precision Points. *J. Symb. Comput.* 45, 1 (2010), 19–37.

[14] Jean-Charles Faugère, Patrizia Gianni, Daniel Lazard, and Teo Mora. 1993. Efficient Computation of Zero-dimensional Gröbner Bases by Change of Ordering. *J. Symb. Comput.* 16 (1993), 329–344.

[15] A. V. Geramita, M. Kreuzer, and L. Robbiano. 1993. Cayley-Bacharach schemes and their canonical modules. *Trans. Amer. Math. Soc.* 339 (1993), 163–189.

[16] Hans-Gert Gräbe. 1988. On Lucky Primes. *J. Symb. Comput.* 6 (1988), 183–208.

[17] Nazeran Idrees, Gerhard Pfister, and Stefan Steidel. 2011. Parallelization of Modular Algorithms. *J. Symb. Comput.* 46 (2011), 672–684.

[18] Martin Kreuzer and Lorenzo Robbiano. 2008. *Computational Commutative Algebra 1* (second ed.). Springer, Heidelberg.

[19] Martin Kreuzer and Lorenzo Robbiano. 2016. *Computational Linear and Commutative Algebra.* Springer, Heidelberg.

[20] Y. Lakshman. 1991. A Single Exponential Bound on the Complexity of Computing Groebner Bases of Zero Dimensional Ideals. In *MEGA 90 – Effective Methods in Algebraic Geometry. Progress in Mathematics*, Teo Mora and Carlo Traverso (Eds.), Vol. 94. Birkhäuser, Boston, 227–234.

[21] Maria Grazia Marinari, H. Michael Möller, and Teo Mora. 1993. Gröbner Bases of Ideals Defined by Functionals with an Application to Ideals of Projective Points. *Appl. Algebra Eng. Commun. Comput.* 4 (1993), 103–145.

[22] Hans Michael Möller. 1998. Gröbner Bases and numerical analysis. In *Gröbner Bases and Applications [Proceedings of the conference 33 years of Gröbner bases]*, Bruno Buchberger and Franz Winkler (Eds.). London Mathematical Society Lecture Note Series, Vol. 251. Cambridge University Press, Cambridge, 159–178.

[23] Michael Monagan. 2004. Maximal Quotient Rational Reconstruction: An Almost Optimal Algorithm for Rational Reconstruction. In *Proc. ISSAC, ACM 2004*. ACM, New York, 243–249.

[24] Masayuki Noro. 2002. An efficient modular algorithm for computing the global b-function. In *Proceeding of the First Congress of Mathematical Software*, Arjeh M Cohen, Xiao-Shan Gao, and Nobuki Takayama (Eds.). World Scientific, 147–157.

[25] Masayuki Noro and Kazuhiro Yokoyama. 1999. A modular method to compute the rational univariate representation of zero-dimensional ideals. *J. Symb. Comput.* 28 (1999), 243–263.

[26] Masayuki Noro and Kazuhiro Yokoyama. 2004. Implementation of prime decomposition of polynomial ideals over small finite fields. *J. Symb. Comput.* 38 (2004), 1227–1246.

[27] Masayuki Noro and Kazuhiro Yokoyama. 2018. Usage of Modular Techniques for Efficient Computation of Ideal Operations. *Math.Comput.Sci.* 12 (2018), 1–32.

[28] Elisa Palezzato. 2017. *Minimal Polynomials, Sectional Matrices, and Applications.* Ph.D. Dissertation. Università degli Studi di Genova.

[29] Franz Pauer. 2007. Gröbner bases with coefficients in rings. *J. Symb. Comput.* 42 (2007), 1003–1011.

[30] Lorenzo Robbiano. 1998. Gröbner Bases and Statistic. In *Gröbner Bases and Applications [Proceedings of the conference 33 years of Gröbner bases]*, Bruno Buchberger and Franz Winkler (Eds.). London Mathematical Society Lecture Note Series, Vol. 251. Cambridge University Press, Cambridge, 179–204.

[31] Shojiro Sakata. 1998. Gröbner Bases and Coding Theory. In *Gröbner Bases and Applications [Proceedings of the conference 33 years of Gröbner bases]*, Bruno Buchberger and Franz Winkler (Eds.). London Mathematical Society Lecture Note Series, Vol. 251. Cambridge University Press, Cambridge, 205–220.

[32] Yao Sun and DingKang Wang. 2013. An efficient algorithm for factoring polynomialsover algebraic extension field. *Sci. China Math.* 56, 6 (2013), 1155–1168.

[33] Franz Winkler. 1988. A p-adic Approach to the Computation of Gröbner Bases. *J. Symb. Comput.* 6 (1988), 287–304.