Primary decomposition of zero-dimensional ideals: Putting Monico's algorithm into practice

Marc Moreno Maza Éric Schost Wenqin Zhou

March 13, 2006

Abstract

Monico published in [*Journal of Symbolic Computation*, 34(5):451–459, 2002] an algorithm to compute the primary decomposition of a zero-dimensional ideal, that mostly relies on a characteristic polynomial computation modulo the input ideal, and its factorization.

We revisit this algorithm, and discuss Maple and Magma implementations that contradict the somehow pessimistic conclusions of Monico's original article: this algorithm provides competitive, sometimes faster alternatives to built-in functions in both systems. We also give an estimation of the probability of success of the algorithm.

Keywords: Primary decomposition, zero-dimensional ideal, Gröbner basis, characteristic polynomial.

1 Introduction

In [9], Monico presents an algorithm for computing the primary decomposition of a zero-dimensional ideal, starting from a Gröbner basis of this ideal (see also further discussions by Cox [4, 3]). However, the experiments in [9], conducted in the Singular computer algebra system, were clearly in favor of the built-in primary decomposition routines, which implement the algorithm of [6]. A conclusion of [9] was thus that ... *"this algorithm, while relatively easy to implement, is only of practical interest if the vectorspace dimension of the quotient ring is small"*.

In this extended abstract, we report on new sets of experiments, in the Maple and Magma systems, for which our implementation of Monico's algorithm competes with, or outperforms, the built-in primary decomposition facilities.

Algorithmically, the main issue is the computation of the characteristic polynomial of an element modulo the input ideal. As in [11], we use a solution to this question relying on trace computations and Newton's formulas. Besides, Monico's algorithm is probabilistic; using classical zero-avoidance results [16, 13], we give estimates on the probability of failure of this algorithm.

2 Description of the algorithm

2.1 Overview

Let I be a zero-dimensional ideal in $k[X_1, \ldots, X_n]$, where k is any field (we denote by $\overline{k}$ one of its algebraic closures). We aim at computing a minimal primary decomposition of I, which we will write

$$I = Q_1 \cap \cdots \cap Q_s.$$

To this effect, Monico's algorithm takes as input a Gröbner basis for the ideal I, and outputs polynomials $R_1, \ldots, R_s$, such that for all $i = 1, \ldots, s$, $Q_i = I + \langle R_i \rangle$ holds. From this, one may compute Gröbner bases for all Q_i, if required.

The polynomials R_i are obtained as follows. Let A be the residue class ring

$$A = k[X_1, \ldots, X_n]/I,$$

and let t be a "generic" element in A (the genericity assumption is discussed more precisely in Subsection 2.3). Write $\chi_t \in k[T]$ for the characteristic polynomial of the endomorphism

$$\mu_t : \begin{array}{ccc} A & \to & A \\ u & \mapsto & ut, \end{array}$$

which we call the *characteristic polynomial of* t. Let finally $c_1, \ldots, c_\ell \in k[T]$ be the irreducible factors of χ_t, and $m_1, \ldots, m_\ell$ their multiplicities. For $i = 1, \ldots, \ell$, define $r_i = c_i^{m_i}(t) \in A$, and take for R_i any preimage of r_i in $k[X_1, \ldots, X_n]$. Then we have the equality (Proposition 2.3 in [9])

$$I = (I + \langle R_1 \rangle) \cap \cdots \cap (I + \langle R_\ell \rangle).$$

Furthermore, as said above, under suitable genericity conditions on t, ℓ equals the number s of primary components of I, and the ideals $I + \langle R_i \rangle$ are then these primary components.

2.2 Details of the implementations

Given the input Gröbner basis, and $t \in A$, the main tasks in this algorithm are:

1. computing the characteristic polynomial χ_t of t;
2. factoring χ_t as $\chi_t = c_1^{m_1} \cdots c_\ell^{m_\ell}$;
3. computing $c_i^{m_i}(t)$ for all i.

In this work, we concentrated on points 1 and 3, leaving factorization to the built-in routines.

Characteristic polynomial computation. Several solutions are available to compute the characteristic polynomial of an element t in A. In our experiments, the most efficient solution turned out to strongly depend on the implementation platform.

Built-in characteristic polynomial routines turned out to be the bottleneck in our preliminary Maple implementation. Our solution for this platform comes from Rouillier's work [11], and is closely related to Leverrier's algorithm [8] (note that primary decomposition is already mentioned as an application of the RUR algorithm in [11]).

Let tr be the *trace* linear form on the quotient A, which maps $t \in A$ to the trace of the map μ_t. The following classical proposition, a consequence of the so-called "Stickelberger Theorem" [5, Proposition 4.2.8] shows how to use this linear form for characteristic polynomial computation.

Proposition 2.1. *For all $t \in A$ and i in $\mathbb{N}$, $\mathrm{tr}(t^i)$ is the ith power sum (Newton sum) of the characteristic polynomial of t.*

Our algorithm for characteristic polynomial computation then follows Rouillier's. Note that as input, we know a Gröbner basis of I, and thus a monomial basis $B = b_1, \ldots, b_D$ of the quotient A. Due to the use of Newton's relations below, we have to assume that D is less than the characteristic of the base field.

1. Compute the matrix M_t of the endomorphism μ_t in the basis $b_1, \ldots, b_D$;
2. Compute the trace form, that is, the trace of all elements $b_1, \ldots, b_D$;
3. Compute the powers $1, t, \ldots, t^D$ by successively applying M_t to the vector $[1, 0, \ldots, 0]^t$;
4. Compute the traces of these vectors using the linearity of the trace;
5. Recover the polynomial χ_t using Newton's relations.

More precisely, we start by computing the multiplication matrices of all variables modulo I, and deduce the whole multiplication table of A by successive multiplications. Computing the multiplication matrix of a variable requires D reductions by the given Gröbner basis. Then, deducing the whole multiplication table requires at most $|B \times B|$ matrix/vector products in size D, hence has cost in $O(D^4)$; note however that $|B \times B|$ may be smaller than D^2. The matrix M_t, as well as the traces of all elements $b_1, \ldots, b_D$ are then deduced from the multiplication table, for $O(D^3)$ operations. Step 3 requires D matrix/vector multiplications, for a cost in $O(D^3)$; Step 4 and Step 5 have cost in $O(D^2)$. See [11] for a similar analysis.

In contrast, in the system Magma, the most efficient approach we found uses the built-in `CharacteristicPolynomial` function (with the default settings). Then, on this platform, we do not need to use explicitly the trace form, and thus, we do not need to know the whole multiplication table of A: only the matrix M_t of the multiplication map μ_t is required.

The element t is then taken as a random polynomial of degree 1: the algorithm is still valid with this restriction (see below), and the computations are substantially faster. We then determine the matrix M_t, and deduce its characteristic polynomial using built-in routines.

Evaluation. The final step of the algorithm consists in evaluating univariate polynomials at the element $t \in A$.

- In our Maple implementation, using Leverrier's idea, the required powers of t are already known, hence only constant multiplications and additions are required to perform the evaluation.
- In the Magma implementation, some more work is required, since only the multiplication matrix M_t of t is known; hence, this evaluation is done through Horner's scheme, using the matrix M_t to perform the successive multiplications by t.

2.3 Probabilistic aspects

The validity of this algorithm depends of the choice of t; we now discuss conditions that guarantee success. Recall that $Q_1 \cap \cdots \cap Q_s$ denotes a minimal primary decomposition of I.

Proposition 2.2. *Suppose that for all points α, β in $V(I)$, $t(\alpha) \neq t(\beta)$. Then the previous algorithm correctly computes the primary decomposition of I.*

Proof. By the Chinese Remainder Theorem, one may write $A = \prod_{i \leq s} A_i$, with $A_i = k[X_1, \ldots, X_n]/Q_i$. To any t in A and $i \leq s$, we associate the characteristic polynomial $\chi_{t,i}$ of the image of t in A_i; in particular, χ_t is the product of the polynomials $\chi_{t,i}$. Proposition 2.3 in [9] states that if the $\chi_{t,i}$ are pairwise coprime, the output is correct. By Stickelberger's theorem, the roots of $\chi_{t,i}$ are the values $t(\alpha)$, for α in $V(Q_i)$, and the conclusion of our proposition follows. □

¿From this proposition, we deduce through standard arguments that using generic elements will guarantee success. We first state a result for arbitrary elements in the quotient, and then its analogue for linear forms.

Corollary 2.3. *Let $b_1, \ldots, b_D$ be the given monomial basis of A and let $d \leq D$ be the number of distinct roots of I.*

- *There exists a non-zero polynomial $\Delta \in \overline{k}[T_1, \ldots, T_D]$ of degree $d(d-1)/2$ such that, if $\Delta(t_1, \ldots, t_D) \neq 0$, then using $t = t_1 b_1 + \cdots + t_D b_D$ yields the primary decomposition of I.*
- *There exists a non-zero polynomial $\Delta' \in \overline{k}[T_1, \ldots, T_n]$ of degree $d(d-1)/2$ such that, if $\Delta'(t_1, \ldots, t_n) \neq 0$, then using $t = t_1 X_1 + \cdots + t_n X_n$ yields the primary decomposition of I.*

Proof. Let $\alpha_1, \ldots, \alpha_d$ be the points in $V(I)$, and write $\alpha_i = \alpha_{i,1}, \ldots, \alpha_{i,n}$. We associate to α_i the linear forms

$$a_i(T_1, \ldots, T_D) = b_1(\alpha_i)T_1 + \cdots + b_D(\alpha_i)T_D \quad \text{and} \quad a'_i(T_1, \ldots, T_n) = \alpha_{i,1}T_1 + \cdots + \alpha_{i,n}T_n.$$

Finally, for i, j in $1, \ldots, d$, with $i \neq j$, we define $c_{i,j} = a_i - a_j$ and $c'_{i,j} = a'_i - a'_j$. Then

$$\Delta = \prod_{1 \leq i < j \leq d} c_{i,j} \quad \text{and} \quad \Delta' = \prod_{1 \leq i < j \leq d} c'_{i,j}$$

satisfy our requirements. □

Using the Zippel-Schwartz lemma [16, 13], we deduce the following probability estimate. We still use the notation of the previous corollary.

Corollary 2.4. *Let $\varepsilon > 0$ and let S be a subset of k of size larger than, or equal to, $d(d-1)/2\varepsilon$.*

- *Suppose that $t_1, \ldots, t_D$ are chosen uniformly at random in S. Then the probability that the algorithm outputs the correct result using $t = t_1 b_1 + \cdots + t_D b_D$ is at least $1 - \varepsilon$.*
- *Suppose that $t_1, \ldots, t_n$ are chosen uniformly at random in S. Then the probability that the algorithm outputs the correct result using $t = t_1 X_1 + \cdots + t_n X_n$ is at least $1 - \varepsilon$.*

3 Experimental results

We finally give our computation tables. Times are given in seconds, and are obtained on a 2.60GHz Pentium 4 processor with 1Gb of RAM. All computations are done modulo the prime $p = 33554393$, which leads, in view of Corollary 2.4 and of the corresponding degrees, to a probability of success of at least 0.9996. Time limits were set to 1000 seconds, and memory limits to 2Gb. Most examples below can be found on the web pages of the test suites [1, 15].

Maple implementation. We first report on the Maple implementation, made under Maple version 10. Our timings include the computation of the initial Gröbner basis, as well as those of all output primary components (first and last timing columns). Strictly speaking, these are not part of Monico's algorithm, especially concerning the final Gröbner bases computations. However, this information is obviously of interest for benchmarking this algorithm, all the more as the built-in primary decomposition routine outputs Gröbner bases for the primary components. The numbers reported in Figure 1 are then as follows:

- vars: number n of variables;
- deg: maximal degree of the input equations;
- D: dimension of the quotient A over k;
- t_1: initial Gröbner basis;

System	vars	deg	D	t_1	t_2	t_3	t_4	t_5	t_6	t_7	Total	Maple
Katsura-7	8	2	128	111	149	3	1	1.2	0.2	970	1235.4	161.2
chemkin	13	3	40	19	18	0.1	0.2	0.1	0.1	5.8	43.3	22.9
Pinchon-2	10	4	48	39.2	20.9	0.2	0.2	0.2	0.1	34.9	66.4	Error
Methan61	10	2	27	24.1	6.2	0.2	0.1	0.1	0.1	10.8	41.6	27
Rose	3	9	136	0.5	2.9	4.1	1.4	1.4	0.2	32.4	42.9	Error
Cyclic 6	6	6	156	32.5	22.1	9.7	2.2	2	0.2	182.5	251.2	48.6
4 body	6	5	138	310	167.6	5.2	1.8	1.6	0.1	441	927.3	412.3
l-4	5	3	243	0.1	2.4	51.2	4.0	7.0	0.2	314.9	379.8	Error
dessin-2	10	2	42	14.2	13.7	0.1	0.2	0.1	0.1	62.1	90.5	18.5
dessin-18-3	8	3	46	6.8	10.7	1.6	1.6	0.1	0.1	22.2	43.1	12.5
gametwo-5	5	4	44	11.8	6	0.2	0.2	0.1	0.1	15.1	33.5	15.6
r-5	5	6	121	0.1	0.7	3.7	1.0	1.0	0.1	54.1	60.7	Error

Figure 1: Maple timings modulo $p = 33554393$.

- t_2: computation of the matrices of multiplication by $X_1, \ldots, X_n$;
- t_3: computation of all multiplication matrices;
- t_4: trace computations, and deduction of the characteristic polynomial;
- t_5: factorization of the characteristic polynomial;
- t_6: evaluation of the polynomials R_i;
- t_7: Gröbner bases of all primary components;
- Total: total time, sum of $t_1, \ldots, t_7$.
- Maple: built-in Maple primary decomposition, using the function `PrimaryDecomposition` of the `PolynomialIdeals` package.

On these examples, our implementation does not quite reach the efficiency of the built-in Maple routine. However, the ratio never becomes exceedingly large, and we expect that a better tuned, lower-level implementation of Monico's algorithm would yield better results.

Remark next that on some of these examples, the built-in routine outputs the error `(in mod/GetAlgExt) only the single algebraic extension case is implemented`. Our algorithm does not require handling algebraic extensions of the base field, and thus avoids these difficulties, while preserving reasonable performances.

Note finally that the implementation relies on two distinct packages: the `Groebner` package for Gröbner bases computations, and the `LinearAlgebra:-Modular` package for all operations on matrices (using encoding of integers mod p by hardware floats). We want to underline that on all these examples, most of the time, by far, is spent on Gröbner-related operations, to compute Gröbner bases, and the matrices of multiplication by the variables $X_1, \ldots, X_n$. All other multiplication matrices are obtained by linear algebra only, which explains why they are often must faster to compute.

System	vars	deg	D	t_1	t_2	t_3	t_4	t_5	t_6	Total	Magma
Katsura-7	8	2	128	0.25	0.35	0.1	0.3	0.1	0.25	1.53	mem. > 2Gb
Katsura-8	9	2	256	2.4	4	0.15	1.2	0.1	9.4	17.25	time > 1000
chemkin	13	3	40	0.1	0.1	0.1	0.1	0.1	0.1	0.6	2.0
Pinchon-2	10	4	48	0.1	0.1	0.1	0.1	0.1	1.0	1.5	mem. > 2Gb
Methan61	10	2	27	0.1	0.1	0.1	0.1	0.1	0.1	0.6	0.5
Rose	3	9	136	0.1	0.1	0.1	0.4	0.1	0.1	0.9	3.4
Cyclic 6	6	6	156	0.2	0.1	0.1	0.1	0.1	1.0	1.6	0.3
4 body	6	5	138	0.4	0.3	0.1	0.5	0.1	1.5	1.9	mem. > 2Gb
l-4	5	3	243	0.1	0.1	0.1	0.7	0.1	2.0	3.1	3.9
dessin-2	10	2	42	0.1	0.1	0.1	0.4	0.1	0.1	0.9	165.2
dessin-18-3	8	3	46	0.1	0.1	0.1	0.1	0.1	0.1	0.6	1.4
gametwo-5	5	4	44	11.8	0.1	0.1	0.1	0.1	0.1	12.3	330.6
r-5	5	6	121	0.1	0.1	0.1	0.2	0.1	0.3	0.9	0.4

Figure 2: Magma timings modulo $p = 33554393$.

Magma implementation. Next, we discuss the results obtained using Magma version 2.11-2. As before, we count the time for computing the initial Gröbner basis, as well as Gröbner bases for all primary components. In Figure 2, all timings have been rounded up to the next integer multiple of 0.1. The legend of this figure is as follows:

- vars, deg, D: as in Figure 1;
- t_1: initial Gröbner basis;
- t_2: computation of the matrix of multiplication by t;
- t_3: computation of the characteristic polynomial;
- t_4: factorization of the characteristic polynomial;
- t_5: evaluation of the polynomials R_i;
- t_6: Gröbner bases of all primary components;
- Total: total time, sum of $t_1, \ldots, t_6$.
- Magma: built-in Magma primary decomposition, using the function `PrimaryDecomposition`.

The results on the Magma platform are more clearly in our favor, since on most examples, the built-in function takes a (sometimes much) longer time, or Magma fails by exhausting all available resources.

For these examples, the Gröbner computations are much faster in Magma than in Maple, which accounts for the large gap observed between the two systems. On the other hand, the linear algebra computations are more balanced, though still in favor of Magma. Indeed,

for both systems, linear algebra mod p is handled by similar techniques of floating-point encoding.

4 Conclusion and future work

Our main goal in this paper was to show that Monico's algorithm does actually provide a practical way to compute primary decompositions, assuming facilities for Gröbner bases computations, and univariate polynomial factorization. We substantiated this statement by means of Maple and Magma implementations, which, although implemented in a high-level environment, compete, and even outperform in some cases, the available routines. Furthermore, we stressed that the choice of implementation techniques is largely influenced by the relative efficiencies of the available tools.

Further developments are possible along these lines. Indeed, the trace computations used above are a special case of the *power projection* problem, which consists in evaluating a linear form at the powers of a given element t in A. This question, and the dual problem of evaluating a univariate polynomial at t, admit more elaborate solutions, using baby steps / giant steps techniques: the use of these techniques for evaluation is due to Paterson and Stockmeyer [10]; applying them to power projection dates to work of Kaltofen and Shoup, see [14, 7] and references therein. In our context, this baby steps / giant steps approach was already used in [2] and [12]. However, as reported in these references, making profit of this idea seems not to be immediate; more work in this direction is thus required.

References

[1] D. Bini and B. Mourrain. Polynomial test suite. `http://www-sop.inria.fr/saga/POL/`

[2] A. Bostan, B. Salvy, and É. Schost. Fast algorithms for zero-dimensional polynomial systems using duality. *Applicable Algebra in Engineering, Communication and Computing*, 14(4):239–272, 2003.

[3] D. A. Cox. Galois theory via eigenvalue methods. In D. Wang and L. Zhi, editors, *SNC 2005*, pages 166–176, 2005.

[4] D. A. Cox. Solving equations via algebras. In *Solving polynomial equations*, volume 14 of *Algorithms Comput. Math.*, pages 63–123. Springer, Berlin, 2005.

[5] D. A. Cox, J. Little, and D. O'Shea. *Using algebraic geometry*. Springer-Verlag, New York, 1998.

[6] P. Gianni, B. Trager, and G. Zacharias. Gröbner bases and primary decomposition of polynomial ideals. *Journal of Symbolic Computation*, 6(2-3):149–167, 1988. Computational aspects of commutative algebra.

[7] E. Kaltofen. Challenges of symbolic computation: my favorite open problems. *Journal of Symbolic Computation*, 29(6):891–919, 2000.

[8] U. J. J. Leverrier. Sur les variations séculaires des éléments elliptiques des sept planètes principales: Mercure, Venus, la terre, Mars, Jupiter, Saturne et Uranus. *J. Math. Pures Appli.*, 4:220–254, 1840.

[9] C. Monico. Computing the primary decomposition of zero-dimensional ideals. *Journal of Symbolic Computation*, 34(5):451–459, 2002.

[10] M. S. Paterson and L. J. Stockmeyer. On the number of nonscalar multiplications necessary to evaluate polynomials. *SIAM Journal on Computing*, 2(1):60–66, March 1973.

[11] F. Rouillier. Solving zero-dimensional systems through the Rational Univariate Representation. *Applicable Algebra in Engineering, Communication and Computing*, 9(5):433–461, 1999.

[12] F. Rouillier. On solving zero dimensional systems with rational coefficients. Submitted to *Journal of Symbolic Computation*, 2005.

[13] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27(4):701–717, October 1980.

[14] V. Shoup. Efficient computation of minimal polynomials in algebraic extensions of finite fields. In *Proceedings of the 1999 International Symposium on Symbolic and Algebraic Computation (Vancouver, BC)*, pages 53–58, New York, 1999. ACM.

[15] J. Verschelde. The test database of polynomial systems.
`http://www.math.uic.edu/~jan/PHCpack/node10.html`

[16] R. Zippel. Probabilistic algorithms for sparse polynomials. In *Symbolic and algebraic computation*, number 72 in Lecture Notes in Computer Science, pages 216–226, Berlin, 1979. Springer. Proceedings EUROSAM '79, Marseille, 1979.

Marc Moreno Maza
ORCCA, University of Western Ontario, London, Ontario, Canada.
`moreno@orcca.on.ca`

Éric Schost
LIX, École polytechnique, 91128 Palaiseau, France.
`Eric.Schost@polytechnique.fr`

Wenqin Zhou
ORCCA, University of Western Ontario, London, Ontario, Canada.
`wzhou7@scl.csd.uwo.ca`