Solving Zero-dimensional Algebraic Systems

D. LAZARD

LITP, Institut Blaise Pascal, Boîte 168, 4, place Jussieu, F-75252 Paris Cedex 05, France

(Received 27 June 1989)

It is shown that a good output for a solver of algebraic systems of dimension zero consists of a family of "triangular sets of polynomials". Such an output is simple, readable and contains all information which may be wanted.

Different algorithms are described for handling triangular systems and obtaining them from Gröbner bases. These algorithms are practicable, and most of them are polynomial in the number of solutions.

1. Introduction

In many computer algebra problems it is difficult to define what a good solution is. Algebraic systems are typical for such a situation: an algebraic system is a finite set of multivariate polynomials over some field K. Solving it means finding the common zeros of the input polynomials, in an algebraic closure of K.

What does "finding" mean? If the solutions are finite in number, it means "giving the list of the solutions" and we are led to a new question: What is a solution?

When the set of solutions is not finite, another question appears. It is no longer possible to list the solutions and we have to describe them in some useful way. The most natural way is to express the variables as functions of some parameters which may be some of the variables. Unfortunately such an expression is generally impossible with rational functions. So we are led to another question: How to describe an infinite set of solutions?

We leave this last question to another paper (Lazard, 1990) and shall restrict ourselves to the simpler and important case of a finite number of solutions. This number may be large and we are faced with a new question which is not very different from the last one: How to describe a large number of solutions?

Most papers on algebraic systems are mainly concerned with algorithms for finding the solutions. Surprisingly, they do not focus on the form of the provided solutions and on above questions, except in most recent papers (Kobayashi et al., 1988; Gianni & Mora, 1987). These ask for solutions in a form which is a special case of what we call below a triangular set of polynomials. Unfortunately, to get such a special form, linear change of variables are needed which do not preserve sparseness. In recent software, Möller (pers. comm.) gives the solutions in the same form as in the present paper, but uses another method for computing them.

In this paper we show that "triangular sets of polynomials" is a good data structure for representing the solutions. We give different algorithms for transforming triangular sets and obtain them from Gröbner bases. They work with the initial set of variables and thus preserve sparseness as far as it is possible.

The algorithm to obtain triangular sets from a Gröbner base for lexicographical ordering is particularly simple. It seems (this has not yet been checked) that it follows from Langemyr (1991) that it is polynomial in the number of solutions.
Thus, the best method for solving zero-dimensional systems appears to be the following: compute the Gröbner base for the degree-reverse-lexicographical ordering by the Buchberger algorithm; deduce from it the Gröbner base for the pure lexicographical ordering by the algorithm of Faugère et al. (1989); then simplify the result as a family of triangular sets.

If input polynomials have degree $d$ in $n$ variables, the whole process is essentially polynomial in $d^n$ (Lazard, 1983; Lakshman, 1990 and 1991; Lazard & Lakshman, 1991; Faugère et al., 1989; Langemyr, 1991). Thus, these algorithms are nearly optimal: the number of solutions may be $d^n$, by the Bezout theorem.

Let us now describe the structure of the paper. We begin with some examples showing what kind of solutions provide some classical algorithms and what kind of solutions would be useful (section 2). Then, in section 3 we present our data structure for describing the solutions, triangular sets of polynomials. Computing modulo, such a set of polynomials is very close to computation in an algebraic field extension; thus, in section 4, we present method D5, which is actually the best tool for such computations. In section 5, we show that the non-unicity of the description of the solution by triangular sets of polynomials is not a drawback, because it is easy to pass from one description to another.

In sections 6 and 7 we show how the triangular sets may be obtained from a Gröbner basis with good efficiency (nearly optimal) in the finite case; we start in section 6 from any Gröbner basis and in section 7 from a Gröbner basis for a lexicographical ordering, with a better algorithm. The latter is intended to be used with the efficient algorithm for changing the ordering described in Faugère et al. (1989). This gives a polynomial algorithm (in some natural meaning) for all the processes of resolution if the set of solutions is finite even at infinity.

In section 8, algorithms are given in order to pass from solutions given as triangular sets to solutions in the special form used in Kobayashi et al. (1988) and in Gianni & Mora (1987); in this form, only the first polynomial in the triangular set (the one which is univariate) is non-linear in its main variable. This form is more suitable for numeric resolution, but needs changes of variable. It is the only place in this paper where changes of variable are needed.

The fact that our main algorithm is polynomial for a natural measure of complexity shows that the method of resolution is nearly optimal. However, in some cases we get a complicated solution where a simpler form exists, and it would be useful to get this simpler solution. This requires, at least, to have information on the Galois structure of the set of solutions; this is far outside the goal of this paper. But the fact that we do not need any change of variable also has the advantage that some information on the Galois structure may remain apparent in the result and is not necessarily hidden by the choice of generic coordinates.

Finally, it should also be noticed that our problem is a special case of a much more difficult problem, the computation of a primary decomposition which is studied in Gianni et al. (1988).

2. Some Examples

In this section we give some simple examples and show how the results of the classical algorithms are not satisfactory. We first compare the raw results of a Gröbner base computation (for two different orderings) and of the Wu Wen-Tsün (1987) algorithm with the simple result which would be the natural solution of a solver.
Consider the following set of polynomials (which becomes a system of equations by equating them the zero); this system and the subsequent one have been suggested by S. Arnborn (Davenport, 1987).

\[ \begin{align*}
&\ a + b + c + d \\
&\ ab + bc + cd + da \\
&\ abc + bcd + cda + dab \\
&\ abcd - 1.
\end{align*} \tag{1} \]

For the lexicographical ordering, its Gröbner basis is

\[ \begin{align*}
&\ a + b + c + d \\
&\ b^2 + 2bd + d^2 \\
&\ bc - bd + c^2d + cd - 2d^2 \\
&\ bd^4 - b + d^5 - d \\
&\ c^2d^2 - c^2d^2 - d^4 + 1.
\end{align*} \tag{2} \]

For the degree ordering, the basis is

\[ \begin{align*}
&\ a + b + c + d \\
&\ b^2 + 2bd + d^2 \\
&\ b^2c^2 + c^2d - bd^2 - d^3 \\
&\ bcd^2 + c^2d^3 - bd^5 - d^4 - 1 \\
&\ bd^4 + d^5 - b - d \\
&\ c^2d^4 + bc - bd + cd - 2d^2 \\
&\ c^3d^2 + c^2d^3 - c - d.
\end{align*} \tag{3} \]

The raw result of the Wu Wen-Tsün algorithm may be (Scratchpad implementation by M. C. Gontard):

If \( a^2(b^2 - a^2) \neq 0 \) then

\[ (d + c + b + a, (b^2 - a^2)c + ab^2 - a^3, -a^2b^4 + (a^4 - 1)b^2 - a^2) \]

else if \(-a^2b + a^3 \neq 0 \) then

\[ (d + c + b + a, (-ab^2 + a^3)c - a^3b + 1, b^2 - a^2, -a^8 + 2a^4 - 1) \]

else if \(-a^2 \neq 0 \) then

\[ (d + c + b + a, -c^2 - 2ac - a^2, -a^2b + a^3, -a^8 + a^4) \]

else 1.

It is clear that solutions (2) and (4) are sufficiently triangular and give the numerical solutions by successively giving values to the variables. But the wanted result is much simpler: it is not difficult to show that the ideal generated by (1) is the intersection of the three ideals

\[ \begin{align*}
&\ (a + c, b + d, c^2d^2 - 1) \\
&\ (a + b + 2d, (b + d)^2, c - d, d^4 - 1) \\
&\ (a - d, b + c + 2d, (c + d)^2, d^4 - 1).
\end{align*} \tag{5} \]
Even simpler, we may remark that the second polynomial of (2) is a square; adding its square root \( b + d \), we get as a new Gröbner base

\[
(a + c, b + d, c^2d^2 - 1)
\]

which generates the radical of the ideal defined by (1).

The members of (5) and (6) are triangular ideals in the sense that the \( i \)th variable may only appear in the first \( i \) polynomials. We will show that each zero-dimensional system (a system with a finite number of solutions) may be solved as a finite union of such triangular ideals and we will provide efficient algorithms for finding such decompositions.

The preceding system is not zero-dimensional. Thus, let us give another less easy example, the same as before but with one more variable:

\[
\begin{align*}
(a + b + c + d + e) \\
ab + bc + cd + de + ea \\
abc + bcd + cde + dea + eab \\
abcd + bcde + cdea + deab + eabc \\
abcde - 1.
\end{align*}
\]

This system has 70 solutions. The Gröbner base is not so easy to compute. In triangular form, the solutions are

\[
\begin{align*}
(a^5 - 1, b^4 + ab^3 + a^2b^2 + a^3b + a^4, c - a^4b^2, d - a^3b^3, \\
e + a^3b^3 + a^4b^2 + b + a) & \quad (20 \text{ solutions}) \\
(a^5 - 1, b - a, c - a, d^2 + 3ad + a^2, e + d + 3a) & \quad (10 \text{ solutions}) \\
(a^5 - 1, b - a, c^2 + 3ac + a^2, d + c + 3a, e - a) & \quad (10 \text{ solutions}) \\
(a^5 - 1, b^2 + 3ab + a^2, c + b + 3a, d - a, e - a) & \quad (10 \text{ solutions}) \\
(a^{10} + 123a^3 + 1, 55b + a^6 + 144a, 55c + a^6 + 144a, \\
55d + a^6 + 144a, 55e - 3a^6 - 377a) & \quad (10 \text{ solutions}) \\
(a^{10} + 123a^3 + 1, 55b - 3a^6 - 377a, 55c + a^6 + 144a \\
55d + a^6 + 144a, 55e + a^6 + 144a) & \quad (10 \text{ solutions}).
\end{align*}
\]

These solutions are rather simple. However, the two last groups of 10 solutions seem to be more involved than the others. It is an artefact due to the choice of the ordering of the variables. In fact, the five groups of 10 solutions are the five circular permutations of solutions of the form

\[
(a, a, a, ar, -a(r+3)) \quad \text{where} \ a^2 = 1 \ \text{and} \ r^2 + 3r + 1 = 0.
\]

The simplification of solutions like the two last groups is a very difficult unsolved problem which hardly depends on the Galois structure of the ideal.
3. Triangular Ideals; Theoretical Results

We have seen in the last section that simple and useful solutions are triangular. We will give a precise meaning to this. However, from now we only consider zero-dimensional systems, i.e. systems with only a finite number of solutions. For non-zero-dimensional systems, such triangular systems may also be defined, and all solutions may be defined in terms of some kind of triangular systems, but things are much more difficult (see Lazard, 1990).

Here we consider systems in \( n \) variables \( X_1, \ldots, X_n \) and order them such that
\[
X_1 < X_2 < \cdots < X_n.
\]

**Definition 1.** The **main variable** of a polynomial is the greatest variable (for the above ordering) which appears in it. A set of \( n \) polynomials is **triangular** if the main variable of the \( i \)-th polynomial is \( X_i \) for \( i = 1, \ldots, n \) and if this polynomial is monic as a polynomial in \( X_i \). The degree of a triangular set is the product of the degrees (in their main variables) of its polynomials.

The following three propositions are corollaries of results in Gianni et al. (1988); they may also be easily deduced from most text books in commutative algebra. For the reader's convenience we give direct proofs, because we have not found any reference where they are explicit.

**Proposition 1.** If \( K \) is a field, any maximal ideal in \( K[X_1, \ldots, X_n] \) has a triangular system of generators.

Let \( I \) be a maximal ideal and \( A \) be the field \( K[X_1, \ldots, X_n]/I \). Let us denote by \( x_i \) the image of \( X_i \) in \( A \) and by \( A_i = K(x_1, \ldots, x_i) \) the subfield of \( A \) generated by \( (x_1, \ldots, x_i) \); thus \( A_i = A_{i-1}(x_i) \) is a simple algebraic extension (\( A_i \) being finitely generated as a ring, is an algebraic extension of \( K \)); let \( P_i \) be the minimal monic polynomial of \( x_i \) with coefficients in \( A_{i-1} \); we have \( A_i = A_{i-1}[X_i]/P_i \) and the elements of \( A_i \) are polynomials in \( X_i \) of degree less than the degree of \( P_i \). Thus an easy recursion shows that \( A = K[X_1, \ldots, X_n]/(P_1, \ldots, P_n) \) and that \( P_i \) is a polynomial in \( X_1, \ldots, X_i \) which is monic in \( X_i \).

**Proposition 2.** Every system with a finite number of solutions (in an algebraic closure of \( K \)) is equivalent to the union of a finite number of triangular systems.

Let \( I \) be the ideal generated by the polynomials of the system. The hypothesis implies that \( I \) is zero-dimensional and that its radical is an intersection of maximal ideals. Thus the zeros of the system are zeros of one of these maximal ideals, and the conclusion follows from Proposition 1.

**Proposition 3.** Let \( (P_1, \ldots, P_n) \) be a triangular set of polynomials. Let \( Q_1 \) be an irreducible factor of \( P_1 \), \( A_1 \) be the field \( K[X_1]/Q_1 \); let \( Q_2 \) be an irreducible factor of \( P_2 \) in \( A_1[X_2] \) and \( A_2 \) be the field \( A_1[X_2]/Q_2 \), and so on. The set \( (Q_1, \ldots, Q_n) \) generates a maximal ideal containing \( (P_1, \ldots, P_n) \) and the set of common zeros of the \( P_i \) is the union of the sets of common zeros of all \( (Q_1, \ldots, Q_n) \).

This is clear from the preceding proposition.

Thus, from a triangular set of polynomials, finding the maximal ideals containing it, is as easy (or as difficult) as factorization in algebraic extensions.
4. Computing Modulo Triangular Ideals; System D5

It is clear that to compute numerically the common zeros of a triangular set is easy: this consists in solving monic univariate polynomials obtained successively by substituting the variables by the roots of the preceding polynomials. Such a resolution is an example of computation with triangular sets of polynomials. On the other hand, field extensions which are not given by a primitive element (which is generally hard to compute) are defined by triangular sets (Proposition 1), and computing in such field extensions means computing modulo triangular sets. These triangular sets are not the more general ones, but computing modulo a general triangular set is not very different from computing in algebraic extensions, as we shall see now.

Let \( P_1, \ldots, P_n \) be a triangular set in \( K[X_1, \ldots, X_n] \). Note that \( P_1, \ldots, P_k \) is a triangular set in \( K[X_1, \ldots, X_k] \) for \( k = 1, \ldots, n \). Let \( A_k := K[X_1, \ldots, X_k]/(P_1, \ldots, P_k) \). The polynomial \( P_{k+1} \) may be viewed as a polynomial in \( A_k[X_{k+1}] \) and \( A_{k+1} \) is isomorphic with \( A_k[X_{k+1}]/P_{k+1} \). We have seen in Proposition 3 that \( A_n \) is a field iff \( P_{k+1} \) is irreducible as a polynomial in \( A_k[X_{k+1}] \) for \( k = 0, \ldots, n-1 \).

Computing in \( A_k \) is very easy and is more or less implemented in most computer algebra systems: the elements of \( A_k \) are represented as polynomials in \( X_1, \ldots, X_n \) of degree in \( X_k \) less than the degree of \( P_k \) (in \( X_k \)), for \( k = 1, \ldots, n \). The \( P_k \) being monic, dividing by them presents no problem and the multiplication in \( A_n \) is a product of polynomials followed by divisions by \( P_n, P_{n-1}, \ldots, P_1 \).

Inversion in \( A_n \) (when it is a field) proceeds as follows: an element \( Q \) of \( A_n \) is a polynomial with \( X_k \) as the main variable; compute the extended GCD of \( Q \) and \( P_k \) in \( A_{k-1}[X_k] \) (this needs inversions in \( A_{k-1} \)); the result is \( D = QR + P_kS \).

If \( D = 1 \), then \( R \) is the inverse of \( Q \); if \( D = P_k \), then \( P_k \) divides \( Q \) and \( Q = 0 \) is not invertible in \( A_k \); there are no other possibilities if \( A_n \) is a field, because \( P_k \) is irreducible. If \( A_n \) is not a field, and \( D \neq 1, D \neq P_k \), then \( P_k \) is a product, and we have found factors without a factorization algorithm. If we replace \( P_k \) by each of its factors (\( D \) and \( P_k/D \)), we get two triangular sets; modulo the first one, \( Q = 0 \) is not invertible; modulo the second one, the inverse of \( Q \) is \( RD^{-1} \); this needs to invert \( D \) modulo \( P_k/D \).

Thus we may compute in \( A_k \) as if it were a field, under the condition of eventually splitting it. This has been remarked by Dominique Duval and implemented by her and Claire Dicrescenzo in REDUCE and SCRATCHPAD II (now called Axiom), under the name D5 (Della Dora et al., 1985; Dicrescenzo & Duval, 1985 and 1988; Duval, 1987). When solving a system of algebraic equations, we want to split it in triangular systems (Proposition 2). This may be done by means of factorization in algebraic extensions (inefficient) or by means of D5. Thus most of the algorithms which follow use D5 or factorization in algebraic extensions, even if we try to avoid them when it is possible.

5. Combining and Splitting Triangular Ideals

**Definition 2.** Two sets of polynomials (and the ideals they generate) are equivalent if they have the same zeros. Two families of sets of polynomials are equivalent if the union of their common zeros are the same.

**Definition 3.** A triangular set \( (P_1, \ldots, P_n) \) of polynomials in \( K[X_1, \ldots, X_n] \) is reduced if \( P_k \) is square-free modulo \( P_1, \ldots, P_{k-1} \) for \( k = 1, \ldots, n \); this means that there exist
polynomials $R$ and $S$ such that $RP_k + SP_k = 1$ modulo $P_1, \ldots, P_{k-1}$ where $P'_k$ is the derivative of $P_k$ with respect to $X_k$.

**Proposition 4.** Any triangular set of polynomials is equivalent with a family of reduced triangular sets, and such a family may be easily computed by $D5$ or by factorization.

A triangular set which generates a maximal ideal being reduced is a corollary of Proposition 3. If $(P_1, \ldots, P_n)$ is the given system, the equivalent family may be obtained by successively replacing $P_i$ by its irreducible factors for $k = 1, \ldots, n$. With $D5$, it suffices to compute, for $k = 1, \ldots, n$, the square-free decomposition of $P_k$ modulo $P_1, \ldots, P_{k-1}$ and to replace $P_k$ by the obtained square-free factors.

Note that computing the square-free decomposition of $P_k$ may induce a splitting of $P_i$ for $i < k$, as is shown by the following example:

\[(P_1 := X_1^2 - X_1; P_2 := X_2^2 + X_1)\]  
\[(10)\]

$P_1$ is a square-free; resultant $(P_2, P'_1) = X_1$, which is 0 if $X_1 = 0$ and 1 is $X_1 = 1$. Thus the equivalent reduced family is $((X_1, X_1), (X_1 - 1, X_1^2 + 1))$. There is no reduced triangular set equivalent to (10). Note also that the family obtained by $D5$ is the same as the family obtained by factorization over the rationals, but not over the Gaussians.

**Proposition 5.** (i) If $(P_1, \ldots, P_n)$ is a triangular system such that $P_k = P'_k \cdot P_k^*$ modulo $(P_1, \ldots, P_{k-1})$ (this means factorization, not derivatives), then $(P_1, \ldots, P_n)$ is equivalent to

\[(((P_1, \ldots, P_{k-1}, P'_k, P_{k+1}, \ldots, P_n), (P_1, \ldots, P_{k-1}, P'_k, P_{k+1}, \ldots, P_n)))\]

where $P'_i$ and $P_k^*$ ($i > k$) are obtained by reducing $P_i$ by $P_1, \ldots, P_{k-1}$ and $P_k$ or $P_k^*$ respectively.

(ii) If $(P_1, \ldots, P_n)$ and $(Q_1, \ldots, Q_n)$ are two triangular sets such that $P_i = Q_i$ for $i < k$, that the degrees of $P_i$ and $Q_i$ are the same for $i > k$, and that $\gcd(P_k, Q_k) = 1$ mod $(P_1, \ldots, P_{k-1})$, then $((P_1, \ldots, P_n), (Q_1, \ldots, Q_n))$ is equivalent to $(P_1, \ldots, P_{k-1}, P_k Q_k, R_{k+1}, \ldots, R_n)$ where the $R_i$ are computed by the Chinese remainder theorem.

Proof is easy. Here are examples of application.

Part (i) may apply to each triangular sets appearing in (8), observing that complete factorization of $a^5 - 1$ is $(a - 1)(a^4 + a^3 + a^2 + a + 1)$ and of $a^{10} + 123a^2 + 1$ is $(a^5 + 3a + 1) \times (a^5 - 3a^2 + 8a^4 - 21a^3 + 55a^2 - 2a + 1)$. Conversely, part (ii) may be used for combining first and fourth, or fifth and sixth triangular sets in (8) to obtain

\[(a^2 - 1, b^2 + 4ab^5 + 5a^2b^4 + 5a^3b^3 + 5a^4b^2 + 4a^5b + a^6,\]
\[\begin{align*}
5c &= 8ab^5 + 30a^2b^4 + 30a^3b^3 + 25a^4b^2 + 30b + 22a, \\
5d &= -2ab^5 - 10a^2b^4 - 15a^3b^3 - 10a^4b^2 - 10b - 8a, \\
5e &= -6ab^5 - 20a^2b^4 - 15a^3b^3 - 15a^4b^2 - 15b - 9a)
\end{align*}\]
\[(a^2 - 1, b - a, c - a, d^2 + 3ad + a^2, e + d + 3a)\]
\[(a^2 - 1, b - a, c^2 + 3ac + a^2, d + c + 3a, e - a)\]
\[(a^2 - 1, b - a, c^2 + 3ac + a^2, d + c + 3a, e - a)\]
\[(a^{10} + 123a^2 + 1, 55b^2 - 2a^2b - 233ab - 8a^2 - 987a^2,\]
\[55c + a^2 + 144a, 55d + a^2 + 144a, 55e + 55b - 2a^2 - 233a)\]

which is the value given by algorithm $D5lexttriangular$ described blow.
We could also combine fifth and sixth triangular sets in (8) and the result with the fourth set of (8). We get another equivalent family which consists of the first three triangular sets of (8) and

\[(a^{12} + 122a^{10} - 122a^5 - 1),\]
\[275b^2 + (16a^{11} + 1958a^6 - 1149a)b + 42a^{12} + 5126a^7 - 4893a^2,\]
\[1375c + (11a^{10} + 1353a^2 + 11)b + 4a^{11} + 517a^6 + 3604a,\]
\[275d - 8a^{11} - 979a^6 + 712a,\]
\[1375e + (-11a^{10} - 1353a^5 + 1364)b + 36a^{11} + 4378a^6 - 5789a).\]

**Remark 1.** It is important to note that the non-uniqueness of the family of reduced triangular sets equivalent to a given system is not a drawback, for the reason that passing from one family to another is rather easy. However, a unique minimal equivalent reduced family does not always exist. For example,
Gröbner($G, K, m, n$): returns the Gröbner bases of the ideal generated by $G$ in $K[X_m, \ldots, X_n]$;
Output: a triangular family equivalent to $G$;
Begin
  if $G$ is triangular then return $[G]$;
  $P := \text{ElimPol}(G, K, m, n)$;
  family := $\{\}$;
  for $Q$ in $\text{Factor}(P, K)$ do
    $L := K[X_m]/Q$;
    $H := \text{Gröbner}(G, L, m+1, n)$;
    $T := \text{Triangular}(H, L, m+1, n)$;
    for $x$ in $T$ replace $x$ by $\text{cons}(Q, x)$;
    family := $\text{append}(T, \text{family})$
  return family
end.

**Proof of the Algorithm.** It follows immediately from the definitions that the roots (in an algebraic closure) of the first elements of a triangular system are the values of the first variable in the solutions. The recursion stops when $m = n$ in the worst case, but, in general, there is only one solution with a given value of $x_1, \ldots, x_k$ for some low value of $k$; in this case, the algorithm stops for $m = k + 1$, the base being triangular (all polynomials in it are linear).

**Remark 2.** Very similar algorithms have been suggested in Lazard (1981) and described in Kobayashi et al. (1988). However, we do not know of any implementation. In fact, this is not easy on most computer algebra systems: we need functions parametrized by a field. Moreover, even if such a parametrization is available as in SCRATCHPAD, we need also a good factorization algorithm on towers of simple algebraic extensions.

**Remark 3.** Many Gröbner bases over field extensions are computed during this algorithm. This may appear to be prohibitive. In fact the polynomials in these bases are generally of very low degree and it seems that in most cases, the total cost of these Gröbner bases computations is dominated by computation of the Gröbner base of the initial ideal. In any case, we give in the following sections a method which avoids multiple Gröbner base computations.

Before describing $\text{ElimPol}$, we give the D5 variant of $\text{Triangular}$.

**Procedure 2** $\text{D5Triangular}(G, \text{mod}, m, n)$

Input:
- **mod**: a triangular family of $m - 1$ polynomials in $x_1, \ldots, x_{m-1}$ which is empty for the main call;
- $m$: current index ($m = 1$ for the main call);
- $n$: the number of variables;
- $G$: the Gröbner bases of polynomials in $x_m, \ldots, x_n$ over the D5-field of the polynomials in $x_1, \ldots, x_{m-1}$ modulo $\text{mod}$;

Output: a triangular family equivalent to $G$;
Auxiliary functions:
- $\text{D5ElimPol}(G, \text{mod}, m, n)$: returns a polynomial in $x_m$ over the D5-field of the polynomials in $x_1, \ldots, x_{m-1}$ modulo $\text{mod}$;
\texttt{DSgröbner}(G, \textit{mod}, m, n): returns the Gröbner bases of the ideal generated by \( G \) over the \( D5 \)-field of the polynomials in \( x_1, \ldots, x_{m-1} \) modulo \textit{mod};

\textit{Begin}

\textbf{If} \( G \) is triangular \textbf{then return} \texttt{append}(\textit{mod}, G);
\( P := \texttt{DSelimpol}(G, \textit{mod}, m, n); \)
\( \textit{mod} := \text{endcons}(P, \textit{mod}); \)
\( G := \texttt{DSgröbner}(G, \textit{mod}, m+1, n); \)
\( \texttt{DStriangular}(G, \textit{mod}, n+1, n); \)
\textit{end.}

This algorithm is exactly the same as \texttt{Triangular}, except that the loop does not appear explicitly but is implicitly created by \( D5 \).

\textbf{Remark 4.} Both \texttt{Triangular} and \texttt{D5triangular} may return non-reduced triangular ideals. This may be avoided by a post-computation of square-free decompositions or by continuing recursive calls until \( m = n \): \( D5 \) system considers only square-free \textit{moduli} (here and in the following, we call moduli the members of a triangular system when computations are done modulo this system); if a modulus is not square-free, it is automatically split by \( D5 \) by square-free decomposition; on the other hand, in \texttt{Triangular}, the loop on the factors removes also the multiplicities.

\textbf{Remark 5.} The algorithm \texttt{DSgröbner} is exactly the standard Buchberger's one managed by \( D5 \). Thus it may split the moduli. It would be possible to replace it by an algorithm without splitting: it is easy to define Gröbner bases over a reduced artinian ring; the computation would be essentially the same as the classical Buchberger's algorithm on the ideal generated by \texttt{append}(\( G \), \textit{mod}) with the following ordering: lexicographical on \( x_1, \ldots, x_{m-1} \) and considering the powers of \( x_1, \ldots, x_{m-1} \) only when the powers of \( x_m, \ldots, x_n \) are the same for the monomials to be compared. However, split problems are generally much easier than the initial problem and it is probably more efficient to use \texttt{DSgröbner}.

We now present two ways for implementing \texttt{ElimPol}. The first one is classical and appears at least in \cite{Kobayashi}. We have

\textbf{Procedure 3 ElimPolMin}(\( G, K, m, n \))

\textbf{Input:} \( G \): a Gröbner base of an ideal in \( K[x_m, \ldots, x_n] \);
\textbf{Output:} the minimal univariate polynomial in \( K[x_m] \) which is in the ideal generated by \( G \);

\textit{Begin}

\( \text{monom} := 1; \text{nfm} := \text{NormalForm}(G, \text{monom}); \)
\( \text{lnf} := []; \text{listmonom} := []; \)
\textbf{while} \text{nfm} \textbf{is not} a linear combination \text{over} \( K \) \text{of elements in} \text{lnf} \textbf{do}

\( \text{lnf} := \text{cons}(\text{nfm}, \text{lnf}); \)
\( \text{listmonom} := \text{cons}(\text{monom}, \text{listmonom}); \)
\( \text{monom} := x_m * \text{monom}; \)
\( \text{nfm} := \text{NormalForm}(G, \text{monom}) \)
\{ We have \( \text{nfm} = \sum_{e \in \text{lnf}} a_e e \) with \( a_e \in K \) \}

\( \text{return monom} - \sum_{e \in \text{lnf}} a_e \text{mon} \) \text{with} \( \text{mon} \) \text{being the element of} \text{list-monom} \text{corresponding to} \text{e} \text{in} \text{lnf} \)
\textit{end.}
As NormalForm is a linear map, it is clear that this algorithm returns the relation of least degree (modulo $G$) between $1, x_m, x_m^2, \ldots$. It is clear that the linear algebra part needs $O(D^3)$ operations. We refer to Faugère et al. (1989) for an implementation of ElimPolMin with $O(D^3)$ field operations and for a complexity analysis; in fact, ElimPolMin is the beginning of NewBase described there; in the generic case where all solutions of the system are simple and have distinct $x_m$-component, ElimPolMin and NewBase become equivalent.

The second implementation of ElimPol is less efficient; however, it may be easier to implement in some Computer Algebra systems. Moreover, it gives information on multiplicities; in most cases, this information permits us to avoid D5-splitting in DStriangular; thus an implementation of it is possible without D5, which works well on most (not too singular) problems.

**Procedure 4 ElimPolChar**($G, K, m, n$)

*Input:* $G$: a Gröbner base of an ideal in $K[x_m, \ldots, x_n]$;

*Output:* a univariate polynomial in $K[x_m]$ of degree $D$;

*Begin*

For each monomial $mon$ which is in normal form relative to $G$, let $\text{NormalForm}(x_m \ast mon, G) = \sum a_{mon,m} \ast m$ (sum over the monomials in normal form);

return the characteristic polynomial of the matrix $(a_{mon,m})$.

*end.*

The matrix constructed in ElimPolChar is that of the product by $x_m$ in the algebra of the polynomials modulo the ideal generated by $G$. The result of ElimPolMin is the minimal polynomial of $1$ for this endomorphism; thus it divides the result of ElimPolChar. Moreover, we have the following.

**Proposition 6.** The multiplicity of a root $x$ of the result of ElimPolChar is the sum of the multiplicities of the solutions of $G$ (viewed as a system of equations) with $x$ as $x_m$-component.

This was essentially proved in Lazard (1981). Here is a direct proof. We may suppose w.l.o.g. that $m = 1$. The ring $A = K[x_1, \ldots, x_n]/G$ is a product of local artinian rings with maximal ideal (over an algebraic closure of $K$) of the form $(x_1 - \alpha_1, \ldots, x_n - \alpha_n)$. The multiplicity of the solution is the multiplicity of the corresponding local ring, that is its length or its dimension as a vector space. If $x_1 - \alpha_1$ is in the maximal ideal of a local artinian ring, some power of it is zero; i.e. there exists a $k$ such that $(x_1 - \alpha_1)^k$ is zero in this local ring. This means that the product of the local rings corresponding to the solutions with first component $x_1$ is exactly the union of the kernels of the endomorphisms “product by a power of $x_1 - \alpha_1$”, that is the multiplicity of $x_1$ as an eigenvalue of the endomorphism “product by $x_1$” in $A$.

This function ElimPolChar is very easy to implement but it needs $O(D^4)$ field operations instead of $O(D^3)$ for ElimPolMin. On the other hand, in most examples, the factors given by the square-free decomposition of the result of ElimPolChar suffice to avoid further splittings by D5. For example, on system (7) they give all the decomposition needed to find the solution (11).
7. From Lexicographical Gröbner Base to Triangular Sets

For computing a triangular family from a lexicographical Gröbner base, the main tool is the following theorem by Gianni (1987) and Kalkbrenner (1987).

**Theorem 1.** Let $G$ be a Gröbner base of a zero-dimensional ideal in $K[x_1, \ldots, x_n]$, for the lexicographical ordering such that $x_1 < x_2 < \cdots < x_n$; suppose that $G$ is sorted by increasing leading monomial. Let $f$ be a ring homomorphism of $K[x_1, \ldots, x_k]$ into a field which maps to 0 the elements of $G$ which depend only on $x_1, \ldots, x_k$. Then the first element of $G$ which is not mapped to 0 is the first one which depends only on $x_1, \ldots, x_{k+1}$ with a leading term (as a polynomial in $x_{k+1}$) not mapped to 0. Moreover, the image by $f$ of this polynomial is the GCD of the images by $f$ of all the elements of $G$ depending only on $x_1, \ldots, x_{k+1}$.

The easiest application of this theorem is for numerically solving systems, i.e. when the target field is an algebraic closure of $K$. But this result works with any target field and makes the computation of an equivalent triangular family very easy and possible without any new computation of a Gröbner base. As before, we give two versions of the corresponding algorithm; the first one use factorizations and works with algebraic extensions as target fields; the second one, using D5, works without any factorization.

**Procedure 5 Lextriangular**

*Input:* $G$: a Gröbner base of a zero-dimensional ideal of polynomials in $x_1, \ldots, x_n$, sorted by increasing leading monomials, for the lexicographical ordering such that $x_1 < \cdots < x_n$;

*Output:* $T$: a triangular family equivalent with $G$;

*Subfunctions:*
- $\text{Reduce}(p, \text{mod})$: reduces $p$ modulo $\text{mod}$;
- $\text{Inverse}(p, \text{mod})$: returns the inverse of $p$ modulo $\text{mod}$;
- $\text{Leading}(p, x)$: returns the leading coefficient of $p$ as a univariate polynomial in $x$;
- $\text{Factor}(p, \text{mod})$: returns the list of irreducible monic factors of $p$ modulo $\text{mod}$

*Begin*

$T := \{[\ ]\}$;

for $i$ := 1 to $n$ do

$H$ := the sublist of the elements of $G$ which depends on $x_i$ but not on $x_{i+1}, \ldots, x_n$;

$U := T$; $T := \{\}$;

for $\text{mod}$ in $U$ do

repeat

$p := \text{first}(H)$;

$H := \text{rest}(H)$;

$q := \text{Leading}(p, x_i)$;

$q := \text{Inverse}(q, \text{mod})$;

until $q \not= 0$;

for $q$ in $\text{Factor}(p, \text{mod})$ do

$T := \text{cons}(\text{cons}(q, \text{mod}), T)$

end.

return($T$)

*End.*

The D5 variant of *Lextriangular* differs only by suppressing the loop on $U$ which is managed by D5 and by replacing the loop on the factors by the reduction of $q \star p$ by $\text{mod}$. 
Procedure 6 D51extriangular

Input: $G$: a Gröbner base of a zero-dimensional ideal of polynomials in $x_1, \ldots, x_n$, sorted by increasing leading monomials, for the lexicographical ordering such that $x_1 < \cdots < x_n$;

Output: $T$: a triangular family equivalent with $G$;

Subfunctions:

- $\text{Reduce}(p, \text{mod})$: reduces $p$ modulo $\text{mod}$;
- $\text{Inverse}(p, \text{mod})$: returns the inverse of $p$ modulo $\text{mod}$ or 0 if $p$ reduces to 0 modulo $\text{mod}$;
- $\text{Leading}(p, x)$: returns the leading coefficient of $p$ as a univariate polynomial in $x$;

Begin

$\text{mod} := \{\}$;

for $i := 1$ to $n$ do

$H$ := the sublist of the elements of $G$ which depends on $x_i$ but not on $x_{i+1}, \ldots, x_n$;

repeat

$p := \text{first}(H);$ 
$H := \text{rest}(H);$ 
$q := \text{Leading}(p, x_i);$ 
$q := \text{Inverse}(q, \text{mod});$

until $q \neq 0;$

$p := \text{Reduce}(p \ast q, \text{mod});$

$\text{mod} := \text{cons}(p, \text{mod})$

return($\text{mod}$)

end.

Both algorithms are directly based on Theorem 1: at each step of the loop on $i$, the polynomials in $x-1, \ldots, x_i$ modulo $\text{mod}$ form a field (or a D5-field) and Gianni-Kalkbrenner theorem applies.

In each step of the loop on the moduli these algorithms need only one inversion of the leading coefficient, one product by the inverse and one factorization or splitting for each element of $G$ which is visited. Let $D$ be the number of monomials which are irreducible by the input Gröbner base; the sum of the degrees of the triangular sets (see Definition 1) in the triangular family which is returned is at most $D$ (equality if all solutions are simple). Thus the time needed by $\text{Lextriangular}$ or $\text{D51extriangular}$ is at most the length of $G$ times the time needed for one inversion and one factorization or product of a polynomial by a constant. The length of $G$ is at most $nD$ (see Faugère et al., 1989), where $n$ is the number of variables; thus the above algorithms are polynomial in $D$ and $n$ provided that the needed arithmetic operations are polynomial in the degrees of the triangular set of moduli and of the polynomial to be inverted or factorized. This has been done in Langemyr (1991).

8. Numeric Resolution

We have seen above that numeric solutions may be easily obtained from triangular sets by solving successive univariate polynomials. In fact, this is not so easy because, after the first step, the coefficients are not integers. This is a problem on some computer algebra systems which only provide a function for computing the roots for integer coefficients. More importantly, finding the roots of a polynomial is a very unstable problem.
Thus it is necessary to solve only univariate polynomials with integer coefficients. This is easy from a triangular system.

The following algorithm starts with a triangular system and returns a list of univariate polynomials \( f_0, \ldots, f_n \) such that the set of the evaluations of \( (f_1, \ldots, f_n) \) at the roots of \( f_0 \) is exactly the set of the common zeros of the triangular set. This algorithm is probably not new, but has to be restated in terms of triangular sets.

**Procedure 7 Generic**

**Input:** \( T = (T_1, \ldots, T_n) \): a triangular reduced set in the variables \( x_1, \ldots, x_n \);

**Output:** \( f = (f_0, \ldots, f_n) \): a list of univariate polynomials as specified above;

**Begin**

\[
f_0 := \text{subst}(z, x_1, T_1);
f_1 := z;
\]

for \( i = 2 \) to \( n \) do

\[
p := \text{subst}([f_1, \ldots, f_{i-1}], [x_1, \ldots, x_{i-1}], T_i);
\]

if \( \text{degree}(p, x_i) = 1 \) then

\[
f_i := p - x_i \quad \{p \text{ is monic}\};
\]

else

for \( \lambda \) in \( 1, -1, 2, -2, \ldots \) repeat

\[
f := \text{subst}(z + \lambda x_i, z, f_0);
q := \text{subst}(z + \lambda x_i, z, p);
r := \text{resultant}(f, q, x_i)
\]

until the last non-constant (in \( x_i \)) subresultant is of degree 1 and the coefficient of \( x_i \) in it has a constant \( \text{GCD} \) with \( r \);

\[
s := \text{this subresultant};
\]

\[
f_0 := r;
f_i := \text{subst}(0, x_i, s)/\text{coeff}(x_i, s) \mod r \quad \{\text{this quotient is congruent to a polynomial modulo } r\};
\]

for \( j = 1 \) to \( i - 1 \) do \( f_j := \text{subst}(z + \lambda f_j, z, f_i) \);

end.

**Proof of the Algorithm.** This algorithm replaces \( x_1 \) by a linear combination of it and the other variables and expresses the variables as a polynomial in this new variable. It is clear that it works if the *until* test eventually succeeds and if the resultant is not 0. If the resultant is 0, then \( f \) and \( r \) have a common zero in \( x_i \) for each value of \( z \); substituting back we get that \( f_0 \) and \( p \) have an infinity of common zeros, which is not possible, because \( f_0 \) does not depend on \( x_1 \) and \( p \) is monic in \( x_i \). Thus the resultant is never 0. If the last non-constant subresultant has a degree greater than 1, or if its leading coefficient has a non-trivial \( \text{GCD} \) with the resultant \( r \), then for some root of \( r \) the \( \text{GCD} \) of \( f \) and \( q \) is of degree at least 2. This may be the case for an infinity of values of \( \lambda \) only if \( f_0 \) and \( p \) have a common non-simple zero. This is in contradiction to the reduced hypothesis on \( T \).

**Remark 6.** In practice, it is much better to factor \( f_0 \) at each step and to split the triangular set being constructed: the lower the degree of \( f_0 \), the faster is the computation and smaller are the coefficients which appear. For the same reason, if the leading coefficient of the subresultant of degree 1 has a common factor with the resultant, this may be used for splitting the problem (for example with D5). The same decomposition comes from the factorization of the resulting \( f_0 \), but an earlier decomposition leads to lower values of \( \lambda \).
PROPOSITION 7. The number of arithmetic operations of algorithm Generic is polynomial in the degree $D$ of the input triangular set.

Clearly, at each step, the degree of $f_0$ is less than $D$. Thus the result follows from the fact that resultant computations and factorizations are polynomial, when we remark that the number of iterations on $A$ is bounded by the number of lines passing through two points among $D$.

REMARK 7. If we take into account the growth of coefficients, it appears that Generic is exponential in $n$. It seems that it is unavoidable: we are faced with the difficulty of computing a primitive element of a field extension.

References


