Improving DISPGB Algorithm Using the Discriminant Ideal

Montserrat Manubens [1], Antonio Montes [1]

Departament de Matemàtica Aplicada II.
Universitat Politècnica de Catalunya. Spain.

Abstract

In 1992, V. Weispfenning proved the existence of Comprehensive Gröbner Bases (CGB) and gave an algorithm to compute one. That algorithm was not very efficient and not canonical. Using his suggestions, A. Montes obtained in 2002 a more efficient algorithm (DISPGB) for Discussing Parametric Gröbner Bases. Inspired in its philosophy, V. Weispfenning defined, in 2002, how to obtain a Canonical Comprehensive Gröbner Basis (CCGB) for parametric polynomial ideals, and provided a constructive method.

In this paper we use Weispfenning's CCGB ideas to make substantial improvements on Montes DISPGB algorithm. It now includes rewriting of the discussion tree using the Discriminant Ideal and provides a compact and effective discussion. We also describe the new algorithms in the DPGB library containing the improved DISPGB as well as new routines to check whether a given basis is a CGB or not, and to obtain a CGB. Examples and tests are also provided.

Key words: discriminant ideal, comprehensive Gröbner bases, parametric polynomial system.
MSC: 68W30, 13P10, 13F10.

* Corresponding author: Antonio Montes.
Email addresses: `montserrat.manubens@upc.edu`, `antonio.montes@upc.edu` (Antonio Montes).
URL: `http://www-ma2.upc.edu/~montes` (Antonio Montes).

[1] Work partially supported by the Ministerio de Ciencia y Tecnología under project BFM2003-00368, and by the Generalitat de Catalunya under project 2001 SGR 00224

1 Introduction

Let $R = k[\overline{a}]$ be the polynomial ring in the parameters $\overline{a} = a_1, \ldots, a_m$ over the field k, and $S = R[\overline{x}]$ the polynomial ring over R in the set of variables $\overline{x} = x_1, \ldots, x_n$. Let $\succ_{\overline{x}}$ denote a monomial order wrt the variables $\overline{x}$, $\succ_{\overline{a}}$ a monomial order wrt the parameters $\overline{a}$ and $\succ_{\overline{xa}}$ the product order. The problem we deal with consist of solving and discussing parametric polynomial systems in S.

Since Gröbner bases were introduced various approaches have been developed for this problem. The most relevant ones are:

- Comprehensive Gröbner Bases (CGB) (We92).
- Specific Linear Algebra Tools for Parametric Linear systems (Si92).
- Dynamic Evaluation (Du95).
- Newton Algorithm with Branch and Prune Approach (HeMcKa97).
- Triangular Sets (Mor97).
- Specialization through Hilbert Functions (GoTrZa00).
- DISPGB Algorithm (Mo02).
- Alternative Comprehensive Gröbner Bases (ACGB) (SaSu03).
- Canonical Comprehensive Gröbner Bases (CCGB) (We03).

This paper describes some improvements made on `DISPGB`. Trying to solve some of the examples given in the references cited above using the improved `DISPGB` has been an interesting challenge (see section 5).

In (We92), Professor Volker Weispfenning proved the existence of a Comprehensive Gröbner Basis CGB wrt $\succ_{\overline{x}}$ for any ideal $I \subset S$ such that for every specialization of the parameters $\sigma_{\overline{a}} : R \rightarrow K'$ extended to $R[\overline{x}] \rightarrow K'[\overline{x}]$, $\sigma_{\overline{a}}(\mathrm{CGB})$ is a Gröbner basis of the specialized ideal $\sigma_{\overline{a}}(I)$. He also provided an algorithm to compute it. There are two known implementations of this algorithm (Pe94; Sc91).

In (Mo95) and (Mo98), A. Montes used classical Gröbner bases theory to study the load-flow problem in electrical networks. V. Weispfenning recommended him to use the Comprehensive Gröbner Basis algorithm (We92; Pe94) for this problem. The use of CGB in the load-flow problem provided interesting information over the parameters, but was rather complicated and not very efficient. Moreover, it was not canonical, i.e. it was algorithm depending.

In Montes (Mo02) provided a more efficient algorithm (`DISPGB`) to Discuss Parametric Gröbner Bases, but it was still non-canonical. `DISPGB` produces a set of non-faithful, canonically reduced Gröbner bases (Gröbner system) in a dichotomic discussion tree whose branches depend on the cancellation of some polynomials in R. The ideas in `DISPGB` however, inspired V. Weispfenning in

(We02; We03) to prove the existence of a Canonical Comprehensive Gröbner Basis (CCGB) as well as to give a method to obtain one.

The main idea for building up the canonical tree is the obtention of an ideal $J \subset R$, structurally associated to the ideal $I \subset S$ and the order $\succ_{\overline{x}}$, which clearly separates the essential specializations not included in the generic case. Let us denote J as the *Weispfenning's discriminant ideal* of $(I, \succ_{\overline{x}})$. In the new Weispfenning's algorithm, J must be computed at the beginning of the discussion using a relatively time-consuming method. The discriminant ideal was one of the lacks of the old `DISPGB` and an insufficient alternative algorithm `GENCASE` was provided.

In this paper we obtain, following Weispfenning, a *discriminant ideal* denoted as N, which can be determined from the data obtained after building the `DISPGB` tree using a less time consuming algorithm and, moreover, we prove that $J \subset N$. We conjecture that $J = N$. We have verified it in more than twenty different examples, and no counter-example has been found. The ideal N allows to rewrite the tree getting a strictly better discussion.

We also prove that for a large set of parametric polynomial ideals (at least for all prime ideals I) the discriminant ideal is principal and in this case we have a unique *discriminant polynomial* to distinguish the generic case from the essential specializations. All the theoretical results commented above are detailed in section 2.

In section 3, we describe the improvements introduced in the algorithms. We have made a complete revision to the old release simplifying the algorithm and highly increasing its speed. New routines `CANSPEC` and `PNORMALFORM` which perform semi-canonical specifications of specializations and reductions of polynomials are given. The algorithm has been completely rewritten and the flow control has been simplified. Further reductions of the tree, eliminating similar brother terminal vertices, have been performed using algorithm `COMPACTVERT`.

Following P. Gianni (Gi87), we are interested in guessing whether some basis of I is a comprehensive Gröbner basis or not, in particular for the reduced Gröbner basis of I wrt the product order $\succ_{\overline{xa}}$. We give, in section 4, a simple algorithm `ISCGB` which uses the `DISPGB` output tree to answer that question. We also give an algorithm `PREIMAGE` to compute a faithful pre-image of the non-faithful specialized polynomials from the reduced bases. This allows to construct a CGB. It will be interesting to compare our CGB with Weispfenning's CCGB when implemented.

Finally, in section 5, we give two illustrative examples and a table of benchmarks for `DISPGB` applied to several parametric systems from which the power of the algorithm is clearly shown.

It is stated in the same section that the new `DISPGB`[2] algorithm is efficient and provides a compact discussion of parametric systems of polynomial equations. An incipient version of it was presented in (MaMo04).

2 Generic Case, Discriminant Ideal and Special Cases

Let $K = k(\overline{a})$ be the quotient field of R and IK the ideal I extended to the coefficient field K. Consider $G = \mathrm{gb}(IK, \succ_{\overline{x}})$, the reduced Gröbner basis of IK wrt $\succ_{\overline{x}}$. As K is a field, G can be computed through the ordinary Buchberger algorithm. The polynomials in G have leading coefficient 1. With this normalization g can have denominators in R. Let $d_g \in R$ be the least common multiple of the denominators of g. To obtain a polynomial in S corresponding to g it suffices to multiply g by d_g. Following Weispfenning (We02; We03), for each $g \in G$ we can obtain a *minimal lifting* of g, $a_g g$, such that $a_g g \in I$ and $a_g \in R$ is minimal wrt $\succ_{\overline{a}}$. Doing this for all $g \in G$ we obtain G', a minimal lifting of G which Weispfenning calls the *generic Gröbner basis* of $(I, \succ_{\overline{x}})$. Of course, $d_g \mid a_g$. We will use a sub-lifting of G, $G'' = \{d_g g \;:\; g \in G\} \subset S$, and this will be our *generic case basis* because it is simpler to compute and corresponds to our standard form of reducing polynomials, as it will be seen in section 3.

We call *singular specialization* a specialization σ for which the set of lpp (leading power products) of the reduced Gröbner basis of $\sigma(I)$ is not equal to the set of $\mathrm{lpp}(G, \succ \overline{x})$.

`DISPGB` builds up a binary dichotomic tree $T(I, \succ_{\overline{x}}, \succ_{\overline{a}})$ branching at the vertices whenever a decision about the cancellation of some $p \in R$ has been taken. Each vertex $v \in T$ contains the pair (G_v, Σ_v). $\Sigma_v = (N_v, W_v)$ is the semi-canonical specification of the specializations in v, where N_v is the radical ideal of the current assumed null conditions (from which all factors of polynomials in W_v have been dropped), and W_v is the set of irreducible polynomials (conveniently normalized and reduced by N_v) of the current assumed non-null conditions. Considering W_v^* the multiplicatively closed set generated by W_v, then $G_v \subset (W_v^*)^{-1}\,(K[\overline{x}]/N_v)$ is the reduced form of the basis of $\sigma(I)$ for the specification of the specializations $\sigma \in \Sigma_v$. At a terminal vertex, the basis G_v is the reduced Gröbner basis of $\sigma(I)$, up to normalization, for all specializations $\sigma \in \Sigma_v$.

Weispfenning (We02) introduces the following ideal associated to each $g \in G$:

$$J_g = \{a \in R \;:\; ag \in I\} = d_g\,(I : d_g g) \bigcap R$$

[2] Release 2.3 of the library `DPGB`, actually implemented in *Maple* and available at the site http://www-ma2.upc.edu/∼montes/

the second formula being computable via ordinary Gröbner bases techniques. Then the radical of their intersection $J = \sqrt{\bigcap_{g\in G} J_g}$ is used to distinguish the generic case in the algorithm. We call J the *Weispfenning's discriminant ideal*. A specialization σ is said to be *essential* (for $I, \succ_{\overline{x}}$) if $J_g \subseteq \ker(\sigma)$ for some $g \in G$.

V. Weispfenning proves the following two theorems:

W1: $J = \bigcap \{\ker(\sigma) \ : \ \sigma \text{ is essential }\}$.
W2: Let σ be an inessential specialization. Then
  (i) $\sigma(G)$ is defined for every $g \in G$ and $\mathrm{lpp}(\sigma(g), \succ_{\overline{x}}) = \mathrm{lpp}(g, \succ_{\overline{x}})$.
  (ii) $\sigma(G)$ is the reduced Gröbner basis of the ideal $\sigma(I)$.

In the `DISPGB` tree $T(I, \succ_{\overline{x}}, \succ_{\overline{a}})$ specializations are grouped into disjoint final cases i by the specification Σ_i, and for all specializations in Σ_i the reduced Gröbner bases have the same set of lpp wrt $\succ_{\overline{x}}$.

Let $1 \le i \le k$ number the terminal vertices. We call *singular cases* the final cases for which $\mathrm{lpp}(G_i, \succ_{\overline{x}}) \neq \mathrm{lpp}(G, \succ_{\overline{x}})$. Let A be the set of indexes of the singular cases:

$$A = \{1 \le i \le k \ : \ \mathrm{lpp}(G_i, \succ_{\overline{x}}) \neq \mathrm{lpp}(G, \succ_{\overline{x}})\}.$$

We denote $\mathbb{V}(I)$ the variety of I and $\mathbb{I}(V)$ the ideal of the variety V. The tree, being dichotomic, provides a partition of $(K')^m$ into disjoint sets of specifications, and thus

$$(K')^m = \bigcup_{i=1}^{k} \left(\mathbb{V}(N_i) \setminus \bigcup_{w\in W_i} \mathbb{V}(w) \right) = U_s \bigcup U_g,$$

where U_s is the set of points $\overline{a} \in (K')^m$ corresponding to singular specifications, i.e.

$$U_s(I, \succ_{\overline{x}}) = \{\overline{a} \in (K')^m \ : \ \sigma_{\overline{a}} \text{ is singular }\} = \bigcup_{i\in A} \left(\mathbb{V}(N_i) \setminus \bigcup_{w\in W_i} \mathbb{V}(w) \right).$$

Theorem 1 *Let us call $N(I, \succ_{\overline{x}}) = \mathbb{I}(U_s)$ the* discriminant ideal*. Then*

$$N(I, \succ_{\overline{x}}) = \bigcap_{i\in A} N_i.$$

This theorem allows to compute N from the output of `BUILDTREE`, i.e. the first tree construction in `DISPGB`. (See section 3).

PROOF. We prove both inclusions:

$\subseteq$: $f(\overline{a}) = 0$ for all $f \in N = \mathbb{I}(U_s)$ and $\overline{a} \in U_s$. Thus $\sigma_{\overline{a}}(f) = 0$ for all $\overline{a} \in U_s$. Taking now $\overline{a}$ such that $\sigma_{\overline{a}} \in \Sigma_i$ this implies that $f \in N_i$. As this can be done for all $i \in A$, it follows that $N \subseteq \bigcap_{i\in A} N_i$.

$\supseteq$: For all $f \in \bigcap_{i\in A} N_i$ and all $\overline{a} \in U_s$ there exists $i \in A$ such that $\sigma_{\overline{a}} \in \Sigma_i$ and, of course, $f \in N_i$. Thus $\sigma_{\overline{a}}(f) = 0$, i.e. $f(\overline{a}) = 0$ for all $\overline{a} \in U_s$. Thus $f \in \mathbb{I}(U_s) = N$.

Before proving the next theorem we need the following

Lemma 2 *Any singular specialization is essential.*

PROOF. Let $\sigma_{\overline{a}}$ be a singular specialization. If it were not essential, by Weispfenning theorem (W2), then the reduced Gröbner basis of $\sigma(I)$ would be the generic basis G, and this contradicts the definition of singular specialization. Thus $\sigma_{\overline{a}}$ must be essential.

Theorem 3 $J \subseteq N$.

PROOF. By Weispfenning's theorem (W1), if $f \in J$ then $f \in \ker(\sigma_{\overline{a}})$ for all essential $\sigma_{\overline{a}}$, and thus $f(\overline{a}) = 0$. So, by lemma 2, $f(\overline{a}) = 0$ for all singular $\sigma_{\overline{a}}$. This implies that $f(\overline{a}) = 0$ for all $i \in A$ and $\sigma_{\overline{a}} \in \Sigma_i$ and thus $f \in \sqrt{N_i} = N_i$. Finally, by proposition 1, $f \in N$.

Conjecture 4 *We formulate two forms*

(*i*) (*Strong conjecture*). *All essential specializations are singular.*
(*ii*) (*Weak conjecture*). $J \supseteq N$.

Proposition 5 *The strong formulation of conjecture 4 implies the weak formulation.*

PROOF. If $f \in N$ then, for all $i \in A$, $f \in N_i$. Thus, $f(\overline{a}) = 0$ for all singular specialization $\sigma_{\overline{a}}$ and, if the strong form of the conjecture is true, then $f(\overline{a}) = 0$ also for all $\sigma_{\overline{a}}$ essential and thus $f \in \ker(\sigma_{\overline{a}})$. So, by Weispfenning's theorem (W1), $f \in J$.

In any case, by definition N is discriminant, i.e. for any $\overline{a} \notin \mathbb{V}(N)$ the Gröbner basis of $\sigma_{\overline{a}}(I)$ is generic, and every singular specification is in $\mathbb{V}(N)$. Thus, what we called minimal singular variety in (Mo02) is described by $\mathbb{V}(N)$. If the strong formulation of the conjecture is true then every specialization σ, for which $N \subset \ker(\sigma)$, is not only essential but also singular and thus the corresponding set of lpp of its reduced Gröbner basis cannot be generic.

We have tested our conjecture in more than twenty examples and we have not found any counter-example of any of the two formulations. Nevertheless the weak formulation is the most interesting one and a failure of the strong formulation would not necessarily invalidate the weak formulation.

In most cases Weispfenning's discriminant ideal J is principal, as states the following

Theorem 6 *If $I \subset S$ is a prime ideal and the generic Gröbner basis G wrt $\succ_{\overline{x}}$ is not* [1], *then the discriminant ideal $J(I, \succ_{\overline{x}})$ is principal and is generated by the radical of the* lcm *of all the denominators of the polynomials in G.*

PROOF. Take $g \in G$. We have $J_g = d_g\,(I : d_g g) \bigcap R$. If $h \in J_g$ then $d_g \mid h$, as $d_g g$ has no common factor with d_g. Thus $d_g g\,(h/d_g) \in I$. By hypothesis, $d_g g \neq 1$ and I is prime. So, as $h/d_g \in R$, we have $h/d_g \notin I$. Thus, necessarily $d_g g \in I$ and $d_g \in J_g$. As $d_g \mid h$ for all $h \in J_g$, it follows that $J_g = \langle d_g \rangle$ is principal. As $J = \sqrt{\bigcap_{g \in G} J_g}$ is the intersection of principal ideals, the proposition follows.

Not only prime ideals have principal discriminant ideals as the next example shows: Take

$$I = \langle ax + y + z + b, x - 1 + ay + z + b, x + y + az + b \rangle.$$

Computing the Gröbner basis of I wrt $\mathrm{lex}(x, y, z, a, b)$ one can see that

$$I = \langle (a + 2)z + b, y - z, x + y + az + b \rangle \cap \langle a - 1, x + y + az + b \rangle$$

and I is not prime. The generic Gröbner basis wrt $\mathrm{lex}(x, y, z)$ is, in this case, $G = [z + b/(a + 2), y + b/(a + 2), x + b/(a + 2)]$. Thus $d_g = a + 2$ for each $g \in G$. For this example it is easy to compute $J = \langle (a + 2)(a - 1) \rangle$ which is still principal even if I is not prime and has a prime component with generic Gröbner basis [1].

It would be interesting to characterize which ideals $I \subset S$ have principal discriminant and which do not. But it is now clear that in the most interesting cases we have principal discriminants. This gives a new insight into our concept of singular variety used in the algorithm (Mo02) in order to understand the parallelism and differences between the new Weispfenning's algorithm (We02; We03) and `DISPGB`, and allows us to improve the old algorithm.

Under that perspective, we have completely revised (Mo02) and obtained a much more efficient and compact discussion. An intermediate version was presented in (MaMo04). We shall describe now the improvements introduced in the new `DPGB` library and refer to (Mo02), where the old `DPGB` is described, for all unexplained details.

3 Improved DISPGB Algorithm

In this section we describe the improvements introduced in `DISPGB` algorithm. Table 1 summarizes the basic differences between old (Mo02) and the new algorithms used in it.

First, we have improved the construction of the discussion tree $T(I, \succ_{\overline{x}}, \succ_{\overline{a}})$ in order to have a simpler flow control and to make it faster by avoiding unnecessary and useless time-consuming computations. In the old algorithm this was done by the recursive routine `BRANCH` which was the unique action of `DISPGB`, but now it is done by `BUILDTREE`. As we explain later, it has been strongly reformed.

Then, `DISCRIMINANTIDEAL` computes the discriminant ideal $N = \bigcap_{i \in A} N_i$ which, as shown in section 2, can be determined from `BUILDTREE` output.

After that, `DISPGB` calls `REBUILDTREE`. This algorithm builds a new tree setting the discriminant ideal N at the top vertex and the generic case at the first non-null vertex labelled as [1] (see figure 1 in section 5.1). The old tree is rebuilt under the first null vertex recomputing the specifications and eliminating incompatible branches. The result is a drastic reduction of branches in the new tree. In the old `DPGB` library, this work was partially done by the external algorithm `GENCASE` which has become useless.

To further compact the tree, a new algorithm `COMPACTVERT` is used. It summarizes brother terminal vertices with the same set of lpp into their father vertex. `COMPACTVERT` is called before and after `REBUILDTREE`. `DISPGB` algorithm is sketched in table 2.

3.1 Building up the Discussion Tree: `BUILDTREE`*.*

We have simplified the flow control from the ancient `DISPGB` and dropped useless operations. Now all the hard work of the discussion is done by the recursive algorithm `BUILDTREE` which replaces the old `BRANCH` routine and makes `NEWVERTEX` useless. The obtained discussion is equivalent to the one given by the old `DISPGB`, but now is more compact.

Routines of the old algorithm	Routines of the new algorithm	Improvements	Obsolete routines

`DISPGB`
  `BRANCH`
	`DISPGB`
	 `BUILDTREE`
	 `DISCRIMINANTIDEAL`
	 `REBUILDTREE`
	 `COMPACTVERT`
		`BUILDTREE` replaces old `BRANCH`.
		Current `DISPGB` includes also rebuilding of the tree (`REBUILDTREE`) and `COMPACTVERT`.
			`GENCASE`

`BRANCH` `NEWVERTEX`	`BUILDTREE`	Better flow control, no incompatible branching.	`BRANCH`
`NEWCOND`	`CONDTOBRANCH`	More robust, ensures no incompatible branches.	`NEWCOND`
`CANSPEC`	`CANSPEC`	Uses radical ideal. More robust.	
-	`PNORMALFORM`	Standard polynomial reduction wrt Σ.	
`CONDPGB`	`CONDPGB`	Uses `CONDTOBRANCH` and Weispfenning's standard pair selection.	
-	`DISCRIMINANTIDEAL`	Determines the discriminant ideal N.	
-	`REBUILDTREE`	Rebuilds the tree starting the discussion with N.	`GENCASE` (external)
-	`COMPACTVERT`	Drops brother terminal vertices with same lpp sets.	

Table 1

It computes the discussion tree faster than the old one because now it assembles the discussion over the coefficients of the current basis in one single algorithm, avoiding unnecessary branching and useless computations.

Given B, a set of polynomials generating the current ideal, `BUILDTREE` takes the current basis B_v at vertex v, specialized wrt the current reduced specification $\Sigma_v = (N_v, W_v)$, builds a binary tree T containing the discussion under vertex v, and stores all the data at the vertices of T. It is a recursive algorithm and substitutes the old `BRANCH` and `NEWVERTEX`. See table 3.

Theorem 16 in (Mo02) still applies to the reformed `BUILDTREE`, thus we can assert the correctness and finiteness of the algorithm.

$T \leftarrow$ **DISPGB**$(B, \succ_{\overline{x}}, \succ_{\overline{a}})$
`Input:`
  $B \subseteq R[\overline{a}][\overline{x}]$: basis of I,
  $\succ_{\overline{x}}, \succ_{\overline{a}}$: termorders wrt the variables $\overline{x}$ and the parameters $\overline{a}$ respectively.
`Output:`
  T: table with binary tree structure, containing (G_v, Σ_v) at vertex v
BEGIN
  $T := \phi$, # global variable
  $v := [\ ]$ # (label of the current vertex)
  $\Sigma := ([\ ], \phi)$ # (current specification)
  `BUILDTREE`(v, B, Σ) # (recursive, stores the computations in T)
  $N :=$ `DISCRIMINANTIDEAL`(T)
  `COMPACTVERT`(T) # (compacts T)
  `REBUILDTREE`(T, N) # (rebuilds T)
  `COMPACTVERT`(T) # (compacts T)
END

Table 2

The most important algorithms used by `BUILDTREE` are commented below.

The algorithm `CONDTOBRANCH` substitutes the old `NEWCOND`. It is used each time that `BUILDTREE` is recursively called and also inside `CONDPGB`, applying it to each new not-reducing-to-zero S-polynomial. This prevents Buchberger algorithm from stopping and saves incompatible branches.

Each time we need to know whether a given polynomial $f \in R$ –for example the lc (leading coefficient) of a new S-polynomial– is zero or not for a given specification, we will reduce it by $\Sigma = (N, W)$ using `PNORMALFORM` and then test whether the remainder is compatible or not with taking it null and non-null for each of the specifications using `CANSPEC`. The whole task is done by `CONDTOBRANCH`. See table 4.

`BUILDTREE` uses a Buchberger-like algorithm –`CONDPGB` (Conditional Parametric Gröbner Basis)– taking the specification into account and intending to determine a specializing Gröbner basis. The basic improvements on `CONDPGB` in the new version are: the call to `CONDTOBRANCH` instead of old `NEWCOND` and improving Buchberger algorithm by considering Weispfenning's normal strategy of pair selection (BeWe93). We do not detail these improvements.

`CANSPEC` has also been modified.

At each vertex v of the tree a pair (G_v, Σ_v) is stored, where $\Sigma_v = (N_v, W_v)$ is a specification of specializations. This means that for all $\sigma \in \Sigma_v$ one has $\sigma(N_v) = 0$ and $\sigma(w) \neq 0\ \forall w \in W_v$. From the geometric point of view, a given $\Sigma = (N, W)$ describes the set of points $\mathbb{V}(N) \setminus (\bigcup_{w \in W} \mathbb{V}(w)) \subseteq (K')^m$.

BUILDTREE(v, B, Σ)
`Input`:
v, the label of the current vertex,
$B \subseteq R[\bar{a}][\bar{x}]$, the current basis,
$\Sigma = (N, W)$ the current reduced specification.
`Output`: No output, but the data are stored in the global tree variable T.

```
BEGIN
  c_f := false
  (c_b, c_d, G, Σ_0, Σ_1):=CONDTOBRANCH(B, Σ)
  IF c_d THEN  # (c_d is true if all lc(g), g ∈ G are decided non-null, false otherwise)
    (c_b, c_f, G, Σ_0, Σ_1):=CONDPGB(G, Σ)
  END IF
  T_v := (G, Σ)  # (Store data in the global tree variable T)
  IF c_f THEN  # (c_f is true if the new vertex is terminal, false otherwise)
    RETURN()
  ELSE
    IF c_b THEN  # (c_b is true if null and non-null conditions are both compatibles)
      BUILDTREE((v, 0), G, Σ_0)
      BUILDTREE((v, 1), G, Σ_1)
    ELSE
      BUILDTREE(v, G, Σ_1)  # (and BUILDTREE continues in the same vertex) [a]
    END IF
  END IF
END
```

[a] In this case, if `CONDPGB` has already started then the list of known S-polynomials reducing to 0 can be kept.

Table 3

By proposition 5 in (Mo02), one can see that $\Sigma = (N, W)$ and $\Sigma' = (\sqrt{N}, W)$ describe equivalent specialization sets. And, by proposition 7, the same happens with $\widetilde{\Sigma} = (\widetilde{N}, \widetilde{W})$, where $\widetilde{N}$ has no factor laying in W and is radical, and $\widetilde{W}$ is the set of the irreducible factors of W with multiplicity one reduced modulus $\widetilde{N}$. So we choose the following representative for the specifications describing equivalent specialization sets:

Definition 7 *We call $\Sigma = (N, W)$ a* reduced specification of specializations *if it is a specification such that*

(*i*) *$\langle N \rangle$ is a radical ideal, and $N = \mathrm{gb}(\langle N \rangle, \succ_{\bar{a}})$,*
(*ii*) *there is no factor of any polynomials in $\langle N \rangle$ laying within W,*
(*iii*) *W is a set of distinct irreducible polynomials not laying within $\langle N \rangle$,*
(*iv*) *$\overline{W}^N = W$.*

$(c_b, c_d, G, \Sigma_0, \Sigma_1) \leftarrow$ **CONDTOBRANCH**(B, Σ)

`Input:`

$B \subseteq R[\bar{a}][\bar{x}]$, the current basis

$\Sigma = (N, W)$ a reduced specification.

`Output:`

G is B reduced wrt Σ,

Σ_1 is the reduced specification for the not null branch

Σ_0 is the reduced specification for the null branch

c_b is true whenever Σ_0 exists, and false otherwise.

c_d is true if all $g \in G$ have $\mathrm{lc}(g)$ decided to not null, and false otherwise.

BEGIN

$G :=$ `PNORMALFORM`(B, Σ)

IF there is $g \in G$ with $l_g = \mathrm{lc}(g)$ not yet decided to not null wrt Σ THEN

$c_d :=$ false

$(t, \Sigma_1) :=$ `CANSPEC`$(N_\Sigma, W_\Sigma \bigcup \{l_g\})$

$(t, \Sigma_0) :=$ `CANSPEC`$(\langle N_\Sigma, l_g \rangle, W_\Sigma)$

IF t THEN $c_b :=$ true ELSE $c_b :=$ false ENDIF

ELSE

$c_d :=$ true

ENDIF

END

Table 4

We must note that the set W is not uniquely determined, as there exist infinitely many polynomials which cannot be null for a given specification. For example, suppose that the current reduced specification is $W = \{a\}, N = [a^2 - 1]$. The condition $a \neq 0$ is compatible with N but is redundant in this case. We can also add to W other polynomials like $a - 2$. Thus there is no unique reduced specification, but our choice is convenient enough. The task of obtaining reduced specifications and testing compatibility of the current null and non-null conditions is done by the reformed `CANSPEC`. See table 5.

Proposition 8 *Given any specification of specializations* $\Sigma = (N, W)$*, if* `CANSPEC`(Σ) *returns* $(t, \widetilde{\Sigma})$ *with* $t =$ *true, then* $\widetilde{\Sigma}$ *is a reduced specification of* Σ *computed in finitely many steps. Otherwise it returns* $t =$ *false and* (N, W) *are not compatible conditions.*

PROOF. At the end of each step N_a is a radical ideal, W_a is a set of irreducible polynomials with multiplicity one reduced wrt N_a, so $\overline{W_a}^{N_a} = W_a$. So, N_b is still radical when the algorithm stops, as N_b is built by dropping from N_a all those factors laying in W_a. If the algorithm returns `true`, as at each completed step (N_b, W_b) satisfies the conditions of definition 7, then the conditions are compatible and $\widetilde{\Sigma}$ is a reduced specification of specializations.

```
(t, Σ̃) ← CANSPEC(Σ)
Input: Σ = (N, W) a not necessarily reduced specification.
Output:
  t: a boolean valued variable.
  Σ̃: a reduced specification if t = true, and φ otherwise (in this case
    incompatible conditions have been found).
BEGIN
  N_a := N,  N_b := √N
  W_a := W,  W_b := the irreducible factors of W without multiplicity and
reduced wrt N_a;
  IF ∏_{q∈W_b} q = 0 THEN RETURN(false,φ) ENDIF
  WHILE (N_a ≠ N_b AND W_a ≠ W_b) DO
    N_a := φ
    FOR p ∈ N_b DO
      p := drop from p all irreducible factors laying in W_b
      IF p = 1 THEN RETURN(false,φ) ENDIF
      Add p into N_a
    END FOR
    W_a := W_b
    N_b := √N_a
    W_b := {irreducible factors of W_a without multiplicity and reduced wrt N_b}
    IF ∏_{q∈W_b} q = 0 THEN RETURN(false,φ) ENDIF
  END WHILE
  Σ̃ := (N_a, W_a)
  RETURN(true, Σ̃)
END
```

Table 5

Otherwise the conditions are not compatible.

Let us now see that this is done in finitely many steps. The algorithm starts with $N_0 = N$. At the next step it computes N_1, and then N_2, etc... These satisfy $N_0 \subseteq N_1 \subseteq N_2 \subseteq \cdots$. By the ACC, the process stabilizes. So, only a finite number of factors can exist, thus dropping factors is also a finite process.

The second necessary task is to reduce a given polynomial in S wrt Σ. This is done in a standard form by `PNORMALFORM`. To eliminate the coefficients reducing to zero for the given specification it suffices to compute the remainder of the division by N, because N is radical. And then, in order to further simplify the polynomials, all those factors lying in W are also dropped from N. See table 6.

Nevertheless, the reduction using `PNORMALFORM` does not guarantee that all the

$\tilde{f} \leftarrow$ **PNORMALFORM**(f, Σ)
`Input`: $f \in R[\bar{x}]$ a polynomial, $\Sigma = (N, W)$ a reduced specification,
`Output`: f reduced wrt Σ
BEGIN
$\widetilde{f}$:= the product of the factors of $\overline{f}^N$ not laying in W, conveniently normalized
END

Table 6

coefficients of the reduced polynomial do not cancel out for any specialization $\sigma \in \Sigma$. To test whether adding a new coefficient to the null conditions is compatible with Σ we need to apply `CONDTOBRANCH`.

Given $f, g \in S$ and Σ we say that their reduced forms f_Σ and g_Σ computed by `PNORMALFORM` are equivalent wrt Σ when $\sigma_{\overline{a}}(f)$ and $\sigma_{\overline{a}}(g)$ are proportional polynomials for every particular specialization $\sigma_{\overline{a}} \in \Sigma$ such that $\sigma_{\overline{a}}(\mathrm{lc}(f_\Sigma)) \neq 0$ and $\sigma_{\overline{a}}(\mathrm{lc}(g_\Sigma)) \neq 0$.

Consider for example, $\Sigma = (N = [ab-c, ac-b, b^2-c^2], W = \phi)$, $f_\Sigma = ax + c^2$, $g_\Sigma = cx + c^2 b$ and $\succ_{\overline{a}} = \mathrm{lex}(a, b, c)$. f_Σ and g_Σ are not identical, but note that they are equivalent. As can be seen in this example `PNORMALFORM` is not always able to reduce them to the same polynomial. Nevertheless, we have the following

Proposition 9 *Given two polynomials $f, g \in S$ then $f_\Sigma \sim g_\Sigma$ wrt Σ iff*

(i) $\mathrm{lpp}(f_\Sigma, \succ_{\overline{x}}) = \mathrm{lpp}(g_\Sigma, \succ_{\overline{x}})$ *and*
(ii) `PNORMALFORM` *applied to* $\mathrm{lc}(g_\Sigma)f_\Sigma - \mathrm{lc}(f_\Sigma)g_\Sigma$ *returns 0.*

PROOF. Obviously if one of both hypothesis fail, the reduced expressions are not equivalent wrt Σ.

On the other side, suppose that (i) and (ii) hold. Then, using order $\succ_{\overline{xa}}$ we have $\overline{\mathrm{lc}(g_\Sigma)f_\Sigma}^N = \overline{\mathrm{lc}(f_\Sigma)g_\Sigma}^N$ by hypothesis (ii). Thus, $\mathrm{lc}(g_\Sigma)(\overline{a})\, f_\Sigma(\overline{x}, \overline{a}) = \mathrm{lc}(f_\Sigma)(\overline{a})\, g_\Sigma(\overline{x}, \overline{a})$, for all specializations in Σ. In particular it also holds for those specializations which do not cancel the leading coefficients of f_Σ and g_Σ. And so, it follows that f_Σ and g_Σ are equivalent wrt Σ.

Thus, `PNORMALFORM` does not obtain a canonical reduction of f wrt Σ, but it can canonically recognize two equivalent reduced expressions.

In many practical computations and after applying these algorithms to a number of cases, we have observed that some discussion trees have pairs of terminal vertices hung from the same father vertex with the same lpp set of their bases. As we are only interested in those bases having different lpp sets, then each of these brother pairs, $\{v_0, v_1\}$, can be merged in one single terminal vertex compacting them into their father v and eliminating the distinction of the latter condition taken in v.

Regarding this construction, we can define a partial order relation between two trees if, in this way, one can be transformed into the other.

Definition 10 *Let S and T be two binary trees. We will say that $S > T$ if*

(i) *T is a subtree of S with same root and same intermediate vertices, and*
(ii) *for each terminal vertex $v \in T$ there is in S either the same vertex $v \in S$ such that $(G_{v_T}, \Sigma_{v_T}) = (G_{v_S}, \Sigma_{v_S})$, or a subtree $\overline{S} \subset S$ pending from vertex $v \in S$ with all its terminal vertices $u \in \overline{S}$ with* $\mathrm{lpp}(G_{u_{\overline{S}}}) = \mathrm{lpp}(G_{v_T})$.

So now, given a discussion binary tree T, we may find the minimal tree $\widetilde{T}$ within the set of all trees which can be compared with T regarding this relation. This is done by a recursive algorithm called `COMPACTVERT`.

Let us just note that the minimal tree will not have any brother terminal vertices with the same lpp sets of their bases.

3.3 Rewriting the Tree with the Discriminant Ideal

The tree T built by `BUILDTREE` can be rebuilt using the discriminant ideal N (see section 2). By theorem W2, if we are given $\sigma_{\overline{a}}$ such that there exists some $\delta \in N$ for which $\sigma_{\overline{a}}(\delta) \neq 0$, then $\sigma_{\overline{a}}(I)$ corresponds to the generic case. Thus, placing N into the top vertex labelled as [ ] in the new tree T', for its non-null son vertex we will have $T'_{[1]} = (G_{[1]}, \Sigma_{[1]})$, where $G_{[1]}$ is the generic basis and $\Sigma_{[1]}$ is a union of specifications from T corresponding to

$$\Sigma_{[1]} = \{\sigma \ : \ \exists\, \delta \in N \text{ such that } \sigma(\delta) \neq 0\}.$$

No other intermediate vertices hang from this side of the top vertex. If the strong formulation of conjecture 4 holds, then no generic cases will hang from the first null vertex.

The subtree under the top vertex hanging from the first null son, for which the choice is $\sigma(N) = 0$, will be slightly modified from the original T. The terminal vertices corresponding to singular cases hanging from it will not be modified as, by construction, for all of them the condition is verified by the corresponding specifications. Thus we can rebuild the tree using the recursive algorithm `REBUILDTREE` which goes through the old tree T and rewrites the new one T'. At each vertex v it tests whether the condition N is already included in N_v. If it is the case, then it copies the whole subtree under it. Otherwise it adds N to the null ideal N_v and calls `CANSPEC` to check whether the new condition is compatible or not. If the condition is compatible then the basis will be reduced using `PNORMALFORM` and the algorithm continues. If it is not, then the recursion stops. This algorithm produces a better new tree with possibly less terminal cases (only generic type cases can be dropped). This reconstruction of the tree is very little time-consuming.

3.4 New Generalized Gaussian Elimination `GGE`

We add here a short description of the improvements on the generalized Gaussian elimination algorithm `GGE`.

We realized, by analyzing the procedure of the old `GGE` (Mo02), that there were some special cases for which we could guess the result of the divisions at each step and thus could be skipped. These improvements halve the computing time.

Even though it is more efficient and faster, `GGE` has become not so useful now because the new improvements in `DISPGB`, detailed above, make, in general, `DISPGB` work faster without using `GGE`. So now, the use of `GGE` within the execution of `DISPGB` is just optional (not used by default). However, it can be very useful for other applications, like in the tensegrity problem shown in section 5, to eliminate some variables and simplify a given basis.

4 Comprehensive Gröbner Basis

In (We02; We03) the main goal is to obtain a Comprehensive Gröbner Basis. With this aim, we have built an algorithm, called `ISCGB`, to test whether a given basis G is a comprehensive Gröbner basis for I or not. It uses `PNORMALFORM` algorithm to specialize G for every terminal case in the discussion tree. Then it checks if $\mathrm{lpp}(\sigma(G))$ includes the set of lpp of the reduced Gröbner basis wrt Σ for every terminal case. If this is true for every final case then `ISCGB` returns `true` otherwise returning `false`.

$\tilde{B} \leftarrow \mathbf{CGB}(B, F)$
`Input:`
$B = \mathrm{gb}(I, \succ_{\overline{xa}})$
$F = \{(G_i, \Sigma_i) \ : \ 1 \leq i \leq k\}$ obtained from `DISPGB`
`Output:` $\tilde{B}$ a CGB of I
BEGIN
$\tilde{B} = B$
$\tilde{F}$ = SELECT cases from F for which `ISCGB`$(B, \succ_{\overline{x}})$ is not a CGB.
WHILE $\tilde{F}$ is non empty DO
TAKE the first case $(G_1, \Sigma_1) \in \tilde{F}$
$\tilde{B} = \tilde{B} \bigcup \{$`PREIMAGE`$(g, \Sigma_1, B) \ : \ g \in G_1\}$
$\tilde{F}$ = SELECT cases from $\tilde{F}$ for which `ISCGB`$(\tilde{B}, \succ_{\overline{x}})$ is not a CGB.
END DO
END

Table 7

The algorithm also informs for which cases a given basis is not a CGB. Thus we can compute pre-images of the polynomials for which B does not specialize to a Gröbner basis and add them to the given basis in order to obtain a Comprehensive Gröbner Basis.

Consider a terminal case (G_v, Σ_v) and $g \in G_v$. To simplify notations we do not consider the subindex v. Let $H_g = \{f_1, \ldots, f_r\}$ be a basis of the ideal $I_g = I \bigcap \langle g, N \rangle$ whose polynomials are of the form $qg + n$, with $q \in S$ and $n \in \langle N \rangle$. I_g contains all the polynomials in I which can specialize to g (for those with $\sigma(q)$ a non-null element of R wrt Σ). Set $f'_i = \overline{f_i}^N$. Obviously, $H'_g = \{f'_1, \ldots, f'_r\}$ is a basis of $\sigma(I_g)$. Using Gröbner bases techniques we can express $g \in \sigma(I_g)$ in the form $g = \sum_i \alpha_i f'_i$ where the α_i's are reduced wrt N, as we are in I_g/N. Then $h = \sum_i \alpha_i f_i$ specializes to g and is a pre-image of g in I. This is used to build algorithm `PREIMAGE` which computes a pre-image of g.

Combining `ISCGB` and `PREIMAGE` we compute a CGB using the algorithm sketched in table 7. Let $B = \mathrm{gb}(I, \succ_{\overline{xa}})$, which is a tentative CGB (FoGiTr01; Ka97), and $F = \{(G_i, \Sigma_i) \ : \ 1 \leq i \leq k\}$ the set of final cases of the discussion tree built up by `DISPGB`. `ISCGB` informs about the polynomials in F which do not have a pre-image in the current tentative CGB. `CGB` algorithm adds pre-images of them until a CGB is obtained. Nevertheless, this construction is not canonical and is much more time-consuming than building up the tree, because it uses the product order $\succ_{\overline{xa}}$ instead of working separately wrt $\succ_{\overline{x}}$ and $\succ_{\overline{a}}$.

5 Examples

We have selected two significative detailed examples. The first one is the classical robot arm, which has a very nice geometrical interpretation, and the second one is the study of a tensegrity problem described by a linear system with the trivial null solution in the generic case which has a non principal discriminant ideal. After that, we outline a table containing some relevant information for several other examples.

5.1 *Simple Robot*

The following system represents a simple robot arm (compare with (Mo02)):

$$
\begin{aligned}
B = [&s_1^2 + c_1^2 - 1, s_2^2 + c_2^2 - 1, l\,(s_1\,s_2 - c_1\,c_2) - c_1 + r,\\
&l\,(s_1\,c_2 + c_1\,s_2) + s_1 - z]
\end{aligned}
$$

Using the orders $\mathrm{lex}(s_1, c_1, s_2, c_2)$ and $\mathrm{lex}(r, z, l)$, respectively for variables and parameters, `DISPGB` produces the following outputs: The discriminant ideal is principal: $N = J = [l\,(z^2 + r^2)]$. The set of final cases expressed in the form $T_i = (G_i, (N_i, W_i))$ is:

$$
\begin{aligned}
T_{[1]} \quad &= ([2\,l\,c_2 + l^2 + 1 - z^2 - r^2, 4\,l^2\,s_2^2 + (l^2-1)^2\\
&\quad -2\,(l^2+1)\,(r^2+z^2) + (z^2+r^2)^2,\\
&\quad 2\,(r^2+z^2)\,c_1 - 2\,z\,l\,s_2 - r\,(r^2+z^2-l^2+1),\\
&\quad 2\,(r^2+z^2)\,s_1 + 2\,l\,r\,s_2 + z\,(l^2-r^2-z^2)], \quad ([\,], \{l\,(r^2+z^2)\})).\\
T_{[0,1,1,1]} &= ([2\,l\,c_2 + l^2 + 1, 4\,(l^2-1)\,r\,c_1 + 2\,z\,l\,s_2 - (l^2-1)\,r,\\
&\quad (l^2-1)^2 - 4\,z^2, 4\,(l^2-1)\,z\,s_1 + (l^2-1)^2 + 4\,z^2],\\
&\quad ([z^2+r^2], \{z, l+1, r, l, l-1\})),\\
T_{[0,1,1,0]} &= ([1], \quad ([z, r], \{l+1, l, l-1\})),\\
T_{[0,1,0,1]} &= ([1], \quad ([l^2-1, r^2+z^2], \{z, l\})),\\
T_{[0,1,0,0]} &= ([l\,c_2 + 1, s_2, s_1^2 + c_1^2 - 1], \quad ([l^2-1, z, r], \{l\})),\\
T_{[0,0,1]} \quad &= ([1], \ ([l], \{r^2+z^2-1\})),\\
T_{[0,0,0]} \quad &= ([s_2^2 + c_2^2 - 1, c_1 - r, s_1 - z], \quad ([l, r^2+z^2-1], \{\ \})),
\end{aligned}
$$

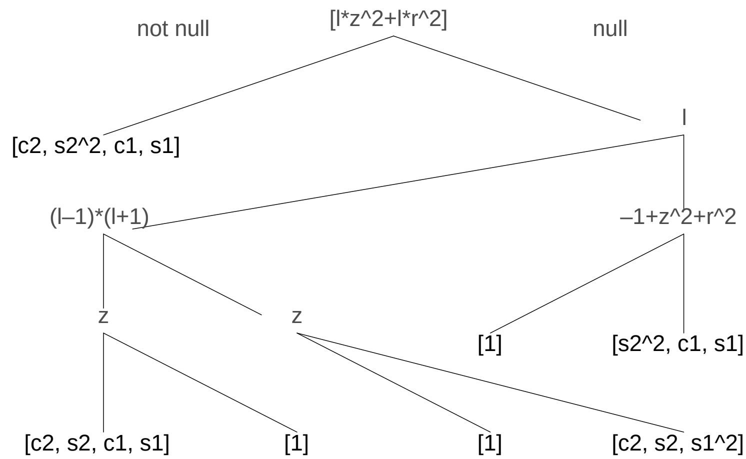

Fig. 1. `DISPGB`'s graphic output for the robot arm.

The generic case $T_{[1]}$ gives the usual formula for the robot. It is characterized by the discriminant ideal N. The singular cases have simple geometrical interpretation and give information about the degenerated cases.

A graphic plot of the tree is also provided in the library. There, the deciding conditions can be visualized at the intermediate vertices and the lpp sets of the reduced Gröbner bases are shown at the terminal vertices (see figure 1).

Now we apply `ISCGB` to $GB = \mathrm{gb}(B, \mathrm{lex}(s_1, c_1, s_2, c_2, r, z, l)$ wrt the output tree. The result is `false`, and the list of specializations for all the final cases is provided:

$$[[1], \{s_1, s_2c_1, s_2s_1, c_1, c_2s_1, c_2, s_1^2, s_2^2\}, \{s_1, c_1, c_2, s_2^2\}, \mathrm{true}]]$$

$$[[0, 1, 1, 1], \{s_1, s_2, s_2c_1, s_2s_1, c_2s_1, c_2, s_1^2, s_2^2\}, \{s_1, s_2, c_1, c_2\}, \mathrm{false}],$$

$$[[0, 1, 1, 0], \{1, s_1, s_2, s_2s_1, c_2s_1, c_2, s_1^2, s_2^2\}, \{1\}, \mathrm{true}],$$

$$[[0, 1, 0, 1], \{s_1, s_2, s_2c_1, s_2s_1, c_2s_1, c_2, s_1^2, s_2^2\}, \{1\}, \mathrm{false}],$$

$$[[0, 1, 0, 0], \{s_2, s_2c_1, s_2s_1, c_2s_1, c_2, s_1^2, s_2^2\}, \{s_2, c_2, s_1^2\}, \mathrm{true}],$$

$$[[0, 0, 1], \{1, s_1, s_2c_1, s_2s_1, c_1, c_2s_1, s_1^2, s_2^2\}, \{1\}, \mathrm{true}],$$

$$[[0, 0, 0], \{s_1, s_2c_1, s_2s_1, c_1, c_2s_1, s_1^2, s_2^2\}, \{s_1, c_1, s_2^2\}, \mathrm{true}],$$

There are only two cases for which GB is not a CGB. Even so, the algorithm `CGB` only needs to add one single polynomial to obtain a CGB.

$$CGB = [2lc_2 + l^2 + 1 - z^2 - r^2,\ c_2^2 + s_2^2 - 1,\ 2(z^2 + r^2)c_1 - 2zls_2$$
$$+r(l^2 - 1 - z^2 - r^2),\ 4zs_2c_1 - 4zrs_2 + 4rc_2c_1 + 4lrc_1$$
$$+2(z^2 - r^2 - 1)c_2 - l(z^2 + r^2 - l^2 + 3),\ 2rc_1s_2 - 2zc_1c_2 - 2zlc_1$$
$$+(-r^2 + z^2 - 1 + l^2)s_2 + 2zrc_2,\ 2(l^2 - 1)s_1 - 4lc_1s_2 + 2ls_2r$$
$$-z(r^2 + z^2 - l^2 - 3),\ 2s_1z + 2c_1r - r^2 - z^2 + l^2 - 1,$$
$$rs_1 - zc_1 + ls_2,\ s_1c_2 + ls_1 - c_1s_2 + rs_2 - zc_2,\ s_1s_2 + c_1c_2$$
$$+lc_1 - zs_2 - rc_2,\ c_1^2 + s_1^2 - 1,\ 4(r^2 + z^2)c_1^2 - 4r(1 + z^2 + r^2 - l^2)c_1$$
$$+(r^2 + z^2 - l^2 + 1)^2 - 4z^2].$$

5.2 Tensegrity Problem

We study here a problem formulated by M. de Guzmán and D. Orden in (GuOr04).

Given the five points $P_1(0,0,0)$, $P_2(1,1,1)$, $P_3(0,1,0)$, $P_4(1,0,0)$, $P_5(0,0,1)$ we want to determine a sixth one $P_6(x,y,z)$ for which the framework with vertices $\{P_1,\dots,P_6\}$ and edges $\binom{\{P_1,\dots,P_6\}}{2}\backslash\{P_1P_6, P_2P_4, P_3P_5\}$ stays in general position and admits a non-null self-stress.

The system describing this problem is the following:

$$B = [w_{12} + w_{14}, w_{12} + w_{13}, w_{12} + w_{15}, w_{12} + w_{23} + w_{25} - w_{26}x + w_{26},$$
$$w_{12} + w_{25} - w_{26}y + w_{26}, w_{12} + w_{23} - w_{26}z + w_{26}, w_{23} + w_{34} + xw_{36},$$
$$w_{13} + w_{34} - w_{36}y + w_{36}, w_{23} + zw_{36}, w_{14} + w_{34} + w_{45} - w_{46}x + w_{46},$$
$$w_{34} + yw_{46}, w_{45} + zw_{56}, w_{15} + w_{45} - zw_{56} + w_{56},$$
$$-w_{26} + w_{26}x + xw_{36} - w_{46} + w_{46}x + w_{56}x,$$
$$-w_{26} + w_{26}y - w_{36} + w_{36}y + yw_{46} + w_{56}y,$$
$$-w_{26} + w_{26}z + zw_{36} + w_{46}z - w_{56} + zw_{56}]$$

Set $\succ_{\overline{xa}} = \mathrm{lex}(w_{12}, w_{13}, w_{14}, w_{15}, w_{23}, w_{25}, w_{34}, w_{45}, w_{26}, w_{36}, w_{46}, w_{56}, x, y, z)$. In order to simplify the system we compute $\mathtt{GGE}(B, \succ_{\overline{xa}})$ (Generalized Gaussian Elimination). The `GGE` basis can be expressed as $B' = B'_1 \cup B'_2$, with B'_2 being the elimination ideal wrt the variables $w_{26} = w_2$, $w_{36} = w_3$, $w_{46} = w_4$,

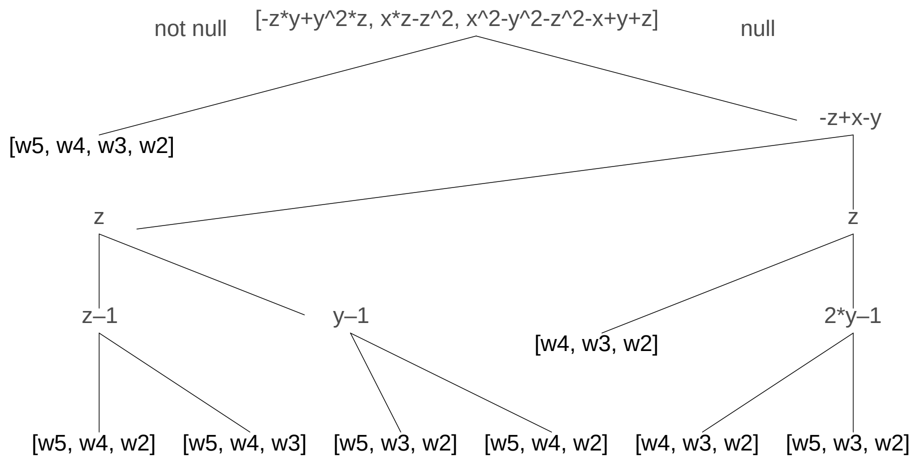

Fig. 2. DISPGB graphic output for the tensegrity problem.

$w_{56} = w_5$, and B'_1 expressing the remaining variables linearly in terms of w_2, w_3, w_4, w_5:

$$\begin{aligned}
B'_1 &= [w_{45} + zw_5, w_{34} + yw_4, w_{25} + w_5 y, w_{23} + zw_3, w_{15} - 2zw_5 + w_5,\\
&\quad w_{14} - 2zw_5 + w_5, w_{13} - 2zw_5 + w_5, w_{12} + 2zw_5 - w_5]\\
B'_2 &= [-zw_5 + w_5 x - w_5 y, -zw_5 + w_4 z, w_4 x + yw_4 - w_4 - zw_5 + w_5,\\
&\quad w_3 y - w_3 + yw_4 - 2zw_5 + w_5, xw_3 - yw_4 - zw_3,\\
&\quad w_2 z - w_2 + zw_3 + 2zw_5 - w_5, w_2 y - w_2 + w_5 y + 2zw_5 - w_5,\\
&\quad w_2 x - w_2 + zw_3 + w_5 y + 2zw_5 - w_5],
\end{aligned}$$

Then, using the orders $\succ_{\overline{x}} = \mathrm{lex}(w_2, w_3, w_4, w_5)$ and $\succ_{\overline{a}} = \mathrm{lex}(x, y, z)$, for variables and parameters respectively, DISPGB produces the following output:

$$\begin{aligned}
T_{[1]} \quad &= ([w_5, w_4, w_3, w_2], ([\ ],\\
&\qquad \{[y^2 z - yz, zx - z^2, x^2 - y^2 - z^2 - x + y + z]\}))\\
T_{[0,1,1,1]} &= ([w_5, w_4, w_2 z - w_2 + zw_3], ([y - 1, x - z], \{z, z - 1\}),\\
T_{[0,1,1,0]} &= ([w_5, w_4, w_3], ([z - 1, y - 1, x - 1], \{\ \})),\\
T_{[0,1,0,1]} &= ([w_5, yw_4 + w_3 y - w_3, w_2], ([z, y - 1 + x], \{2y - 1, y - 1\}),\\
T_{[0,1,0,0]} &= ([w_5, w_4, w_2], ([z, y - 1, x], \{\ \})),\\
T_{[0,0,1]} \quad &= ([-w_5 + w_4, w_3 + 2zw_5 - w_5, w_2 - 2zw_5 + w_5],\\
&\qquad ([y, x - z], \{z\})),
\end{aligned}$$

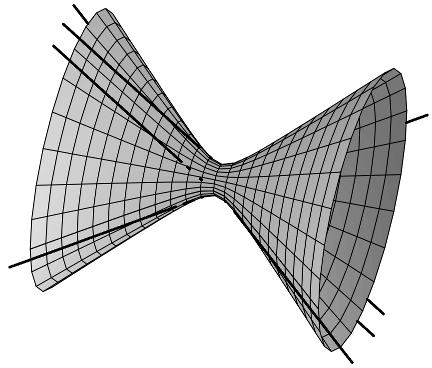

Fig. 3. Location of the sixth point for non null self-stress.

$$
\begin{aligned}
T_{[0,0,0,1]} &= ([w_5 + 2yw_4 - w_4, 2w_3y - w_3 + w_5, w_2 + w_5],\\
&\quad ([z, x - y], \{2y - 1\})),\\
T_{[0,0,0,0]} &= ([w_5, w_3 - w_4, w_2], ([z, 2y - 1, 2x - 1], \{\ \})),
\end{aligned}
$$

and the discriminant ideal is not principal:

$$N = J = [y^2z - yz, zx - z^2, x^2 - y^2 - z^2 - x + y + z].$$

The generic solution is trivial ($w_5 = w_4 = w_3 = w_2 = 0$). In this problem, the interesting non trivial solutions are given by the conditions over the parameters described by the variety of the discriminant ideal, which decomposes into 4 straight lines included in the hyperboloid $x^2-y^2-z^2-x+y+z = 0$ (illustrated in figure 3):

$$\mathbb{V}(N) = \mathbb{V}(z, x - y) \bigcup \mathbb{V}(y, x - z) \bigcup \mathbb{V}(z, x + y - 1) \bigcup \mathbb{V}(y - 1, x - z).$$

For this problem the Gröbner basis wrt variables and parameters is already a comprehensive Gröbner basis.

5.3 Benchmarks

For a set of examples taken from the literature we have applied the current implementation, release 2.3 in *Maple 8* , of algorithm `DISPGB` using a 2 GHz Pentium 4 at 512 MB. Table 8 summarizes the computing time of `DISPGB`, the total number of terminal vertices of the output tree, whether the discriminant ideal is principal or not, and whether the D-Gröbner basis wrt $\succ_{\overline{xa}}$ is a CGB or not, joint by the number of failure cases for which it is not (0 if it is). The

Identification	CPU time (seconds)	# Final vertices	Discriminant is principal?	Is CGB? (# failures)
S1 (We03)	0.8	2	N	Y (0)
S2. (Gi87)	1.2	2	Y	N (1)
S3. (Gom02; Du95)	1.5	2	Y	Y (0)
S4.	1.6	2	N	Y (0)
S5. (Kap95)	1.6	3	Y	N (1)
S6. (Kap95)	2.0	4	Y	Y (0)
S7. (Kap95)	3.0	2	Y	Y (0)
S8. (SaSuNa03)	4.4	3	Y	Y (0)
S9. Similar to (Si92)	6.7	10	Y	Y (0)
S10. Subsection 5.1 Simple robot	7.9	7	Y	N (2)
S11. (Co04) Singular points	8.0	6	Y	Y (0)
S12. (Ry00) Rychlik robot	8.2	11	Y	N (1)
S13. (SaSu03)	8.2	10	Y	Y (0)
S14. (GoTrZa00; De99) ROMIN robot	9.6	2	Y	N (1)
S15. (GoRe93)	18.2	17	Y	N (2)
S16. (GuOr04) Subsection 5.2	21.3	8	N	Y (0)

Table 8

bases of the different examples are detailed below:

- S1. $[a(x+y), b(x+y), x^2+ax]$;
- S2. $[x_1^2, x_1x_2, x_1x_3^2, x_1a+x_2, x_2x_3-x_3^2, x_2a, x_3^3, x_3^2a, a^2]$;
- S3. $[x^3-axy, x^2y-2y^2+x]$;
- S4. $[ax+y-1, bx+y-2, 2x+ay, bx+ay+1]$;
- S5. $[x_4-(a_4-a_2), x_1+x_2+x_3+x_4-(a_1+a_3+a_4), x_1x_3x_4-a_1a_3a_4,$ $x_1x_3+x_1x_4+x_2x_3+x_3x_4-(a_1a_4+a_1a_3+a_3a_4)]$;
- S6. $[vxy+ux^2+x, uy^2+x^2]$;

- S7. $[y^2 - zxy + x^2 + z - 1, xy + z^2 - 1, y^2 + x^2 + z^2 - r^2]$;
- S8. $[a - b + (xya - x^2yb - 3a)^3 + (xyb - 3xb - 5b)^4, xya - x^2yb - 3a,$
  $xyb - 3xb - 5b]$;
- S9. $[x + cy + bz + a, cx + y + az + b, bx + ay + z + c]$;
- S10. See subsection 5.1;
- S11. $[(d_4d_3R + r_2^2 - d_4d_3r_2^2 + d_4^2d_3^2 - d_4d_3^3 - d_4^3d_3 + d_4d_3 + Z - R)t^4$
  $+ (-2r_2d_4R + 2r_2d_4^3 + 2r_2d_4d_3^2 - 4r_2d_3d_4^2 + 2r_2^3d_4 + 2r_2d_4)t^3$
  $- (2r_2^2 - 2R + 4d_4^2r_2^2 + 4d_4^2 + 2Z - 2d_4^2d_3^2)t^2$
  $+ (-2r_2d_4R + 2r_2d_4d_3^2 + 2r_2d_4 + 2r_2d_4^3 + 4r_2d_3d_4^2 + 2r_2^3d_4)t$
  $+ r_2^2 + d_4^3d_3 - d_4d_3R + d_4d_3r_2^2 + Z - R - d_4d_3 + d_4^2d_3^2 + d_4d_3^3]$;
- S12. $[a - l_3c_3 - l_2c_1, b - l_3s_3 - l_2s_1, c_1^2 + s_1^2 - 1, c_3^2 + s_3^2 - 1]$;
- S13. $[ax^2y + a + 3b^2, a(b - c)xy + abx + 5c]$;
- S14. $[t^3 - cut^2 - uv^2 - uw^2, t^3 - cvt^2 - vu^2 - vw^2, t^3 - cwt^2 - wu^2 - wv^2]$;
- S15. $[a + ds_1, b - dc_1, l_2c_2 + l_3c_3 - d, l_2s_2 + l_3s_3 - c, s_1^2 + c_1^2 - 1, s_2^2 + c_2^2 - 1,$
  $s_3^2 + c_3^2 - 1]$;
- S16. See subsection 5.2.

We have tested several other problems and in some of them only partial results have been reached. We detail two significative examples:

- S17. $[axt^2 + bytz - x(x^2 + cy^2 + dz^2), ayt^2 + bzxt - x(y^2 + cz^2 + dx^2),$
  $azt^2 + bxyt - x(z^2 + cx^2 + dy^2)]$
- S18. $[(3x^2 + 9v^2 - 3v - 3x)t_1^2t_2^2 + (3v^2 - 3v + 6vx - 3x + 3x^2)t_2^2$
  $+ (3v + 3v^2 + 3x^2 - 3x - 6vx)t_1^2 - 24v^2t_1t_2 + 9v^2 - 3x + 3x^2 + 3v,$
  $(3x^2 + 9v^2 - 3v - 3x)t_2^2t_3^2 + (3v + 3v^2 + 3x^2 - 3x - 6vx)t_2^2$
  $+ (3v^2 - 3v + 6vx - 3x + 3x^2)t_3^2 - 24v^2t_2t_3 + 9v^2 - 3x + 3x^2 + 3v,$
  $(3x^2 + 9v^2 - 3v - 3x)t_3^2t_1^2 + (3v^2 - 3v + 6vx - 3x + 3x^2)t_1^2$
  $+ (3v + 3v^2 + 3x^2 - 3x - 6vx)t_3^2 - 24v^2t_3t_1 + 9v^2 - 3x + 3x^2 + 3v]$

For S17 (GoTrZa00), `DISPGB` gets bogged down after computing 35 terminal vertices in 1375 sec. It has been unable to finish the tree, and so neither rebuilding with the discriminantideal nor reducing the tree can have been achieved. The label of the 35th vertex is $[1, 1, 0, 1, 0, 0]$, thus all vertices beginning with $[0, 0, \ldots$ have been already determined (the tree is built up in pre-order beginning with the 0 vertices).

S18 corresponds to the benzene molecule studied in (Em99). The situation is similar to S17, getting bogged down after 45 seconds when the 9th vertex labelled $[1, 1, 0, 0]]$ has been computed.

6 Acknowledgements

We want to thank Professor Volker Weispfenning for his useful suggestions and ideas and for encouraging us in undertaking the research on Comprehensive Gröbner Bases, as well as for his hospitality and kindness on the occasions in which we met.

We would like to thank Professor Pelegrí Viader for his many helpful comments and his insightful perusal of our first draft.

We will also thank the referees for their valuable suggestions.

References

[BeWe93] T. Becker, V. Weispfenning. Gröbner Bases: A Computational Approach to Commutative Algebra. Springer, New-York, 1993.

[Co04] M. Coste. Classifying serial manipulators: Computer Algebra and geometric insight. Plenary talk. (Personal communication). *Proceedings* of EACA-2004, 323–323, 2004.

[De99] S. Dellière. Triangularisation de systèmes constructibles. Application à l'évaluation dynamique. Thèse Doctorale, Université de Limoges. Limoges, 1995.

[Du95] D. Duval. Évaluation dynamique et clôture algébrique en Axiom. *Journal of Pure and Applied Algebra*, **99**:267–295, 1995.

[Em99] I. Z. Emiris. Computer Algebra Methods for Studying and Computing Molecular Conformations. *Algorithmica* **25**: 372-402, 1999.

[FoGiTr01] E. Fortuna, P. Gianni and B. Trager. Degree reduction under specialization. *Jour. Pure and Applied Algebra*, **164**(1-2):153–164, 2001. *Proceedings* of MEGA 2000.

[Gi87] P. Gianni. Properties of Gröbner bases under specializations. In: EUROCAL'87. Ed. J.H. Davenport, Springer LCNS, **378**:293–297, Mars 1987.

[Gom02] T. Gómez-Díaz. Dynamic Constructible Closure. *Proceedings* of Posso Workshop on Software, Paris, 73–93, 2000.

[GoRe93] M.J. González-López, T. Recio. The ROMIN inverse geometric model and the dynamic evaluation method. In: Computer Algebra in Industry. Ed. A.M. Cohen, Wiley & Sons, 117–141, 1993.

[GoTrZa00] M.J. González-López, L. González-Vega, C. Traverso, A. Zanoni. Gröbner Bases Specialization through Hilbert Functions: The Homogeneoas Case. *SIGSAM BULL*, Issue 131, **34**:1, 1-8, 2000.

[GuOr04] M. de Guzmán, D. Orden. Finding tensegrity structures: geometric and symbolic aproaches. *Proceedings* of EACA-2004, 167–172, 2004.

[HeMcKa97] P. Van Hentenryck, D. McAllester and D. Kapur. Solving polynomial systems using a branch and prune approach. *SIAM J. Numer. Anal.*, **34** 2:797–827,1997.

[Ka97] M. Kalkbrenner. On the stability of Gröbner bases under specializations. *Jour. Symb. Comp.*, **24**(1):51–58, 1997.

[Kap95] D. Kapur. An Approach for Solving Systems of Parametric Polynomial Equations. In: Principles and Practices of Constraints Programming. Ed. Saraswat and Van Hentenryck, MIT Press, 217–244. 1995.

[MaMo04] M. Manubens, A. Montes. Improving DPGB algorithm for parametric Gröbner basis. *Proceedings* of EACA-2004, 207–211, 2004.

[Mo95] A. Montes. Solving the load flow problem using Gröbner bases. *SIGSAM Bull.*, **29**:1–13, 1995.

[Mo98] A. Montes. Algebraic solution of the load-flow problem for a 4-nodes electrical network. *Math. and Comp. in Simul.* **45**:163–174, 1998.

[Mo02] A. Montes. New algorithm for discussing Gröbner bases with parameters. *Jour. Symb. Comp.*, **33**(1-2):183–208, 2002.

[Mor97] M. Moreno-Maza. Calculs de Pgcd au-dessus des Tours d'Éxtensions Simples et Résolution des Systèmes d'Équatoins ALgebriques. Doctoral Thesis, Université Paris 6, 1997.

[Pe94] M. Pesh. Computing Comprehesive Gröbner Bases using MAS. User Manual, Sept. 1994.

[Ry00] M. Rychlik. Complexity and Applications of Parametric Algorithms of Computational Algebraic Geometry. In: Dynamics of Algorithms. Ed. R. del la Llave, L. Petzold, and J. Lorenz. IMA Volumes in Mathematics and its Applications, Springer-Verlag **118**:1–29, 2000.

[SaSu03] Y. Sato and A. Suzuki. An alternative approach to Comprehensive Gröbner Bases. *Jour. Symb. Comp.*, **36**:649–667, 2003.

[SaSuNa03] Y. Sato, A. Suzuki, K Nabeshima. ACGB on Varieties. *Proceedings* of CASC 2003. Passau University, 313–318, 2003.

[Sc91] E. Schönfeld. Parametrische Gröbnerbasen im Computeralgebrasystem ALDES/SAC-2. Dipl. thesis, Universität Passau, Germany, May 1991.

[Si92] W. Sit. An Algorithm for Solving Parametric Linear Systems. *Jour. Symb. Comp.* **13**:353–394, 1992.

[We92] V. Weispfenning. Comprehensive Gröbner Bases. *Jour. Symb. Comp.* **14**:1–29, 1992.

[We02] V. Weispfenning. Canonical Comprehensive Gröbner bases. *Proceedings* of ISSAC 2002. ACM-Press, 270–276, 2002.

[We03] V. Weispfenning. Canonical Comprehensive Gröbner bases. *Jour. Symb. Comp.* **36**:669–683, 2003.