Improving DisPGB algorithm for parametric Gröbner bases

Montserrat Manubens, Antonio Montes*

Abstract

We present important improvements and a thorough redesign of the algorithm DisPGB in the new release 2.0 of the `DPGB` *Maple* library for discussing Gröbner bases with parameters. `DisPGB20` provides a more compact tree discussion, avoiding incompatible branches, and producing simpler output bases. The new software is more efficient and robust and can increase the speed up to 20 times with respect to the old release.
Keywords: Parametric Gröbner basis, specializations, specification of specializations.

Introduction

Since Gröbner bases were introduced, various Computer Algebra methods for discussing polynomial systems with parameters have been developed. The most relevant ones are:

- Comprehensive Gröbner bases (CGB) [9].
- The Dynamic Evaluation Method [3].
- The Newton algorithm with branch and prune approach [6].
- The DisPGB algorithm [7].
- Alternative Comprehensive Gröbner bases (ACGB) [8].
- Canonical Comprehensive Gröbner bases (CCGB) [10].

Among these methods, DisPGB stands out for being a quite efficient algorithm. Theoretically, the best results should be obtained with CCGB, which seems promising enough. Nevertheless, CCGB is very complex and has not yet been implemented.

The library `DPGB` for discussing Gröbner bases with parameters, implementing the DisPGB algorithm, was described in [7]. Since then, it has been applied successfully to many problems [2]. Applications suggested that it could be –and needed to be– improved. This has been done in the new release `DPGB 2.0` presented here. The new algorithm –from now on denoted DisPGB20– provides the following improvements:

- a much simpler flow control of the recursive algorithm;
- a better `CANSPEC` algorithm to determine the semi-canonical specification of the specializations, that uses zero-dimensional radical computation;
- saves incompatible branches and unnecessary stops and restarts of the algorithm;
- saves computations already done in previous calls that are useful in the actual branch;

*Partially supported by the Ministerio de Ciencia y Tecnología under project BFM2003-00368 and by the Generalitat de Catalunya under project 2001 SGR 00224

- carries out canonical simplification of the polynomials at each step and at the final presentation of the basis, providing much better results and more efficient computations.
- new improvement of the `GGE` routine that increases its speed by a factor 2;
- improvements in the output routines `tplot` and `finalcases`.

The result is a more robust and efficient library that is freely available at the web [1]. It increases the speed up to 20 times in most examples and allows computations that were not possible with the old `DPGB`.

1 The new DisPGB20 algorithm

Suppose we are given a basis, F, of an ideal and a specification of specializations, summarized in the null and not-null conditions, $\Sigma = (N, W)$. The main idea of the DisPGB algorithm lays on discussing the nullity or not wrt Σ of the leading coefficients, say $\mathrm{lc}(f)$, of polynomials appearing at each step. We do this in a recursive routine, say `CONDPGB`, in which the discussion for the polynomials given in the initial basis and for the polynomials appearing in the Buchberger algorithm takes place. Let's see:

For each $f \in F$, we test whether $\mathrm{lc}(f)$ lays in $\sqrt{\langle N\rangle}$, in W, or none of them. Whenever $\mathrm{lc}(f) \in \sqrt{\langle N\rangle}$, as this implies that $\mathrm{lc}(f)$ specializes to zero, we can "radicalize" N adding $\mathrm{lc}(f)$ to it, and then make the proper reductions in W and N with the improved `CANSPEC`, so that Σ gets semi-canonical. At the same time, we substitute f by $f' = f - \mathrm{lm}(f)$ and then start again. If $f' = 0$ then we remove f' from the basis and go on with the next polynomial.

If we reach some f' such that $\mathrm{lc}(f') \notin \sqrt{\langle N\rangle}$ and $\mathrm{lc}(f') \notin W$, what we have to do is making a new assumption about $\mathrm{lc}(f')$ and splitting the discussion into two supplementary cases: assuming $\mathrm{lc}(f') \neq 0$ and assuming $\mathrm{lc}(f') = 0$, so we will have two new specifications of specializations. But before making these assumptions, we must ensure they are compatible with current null and not null conditions (N, W). If no incompatibility is found, we obtain $\Sigma_1 =$`CANSPEC`$(N, W \cup \{\mathrm{lc}(f')\})$ and $\Sigma_0 =$`CANSPEC`$(N \cup \{\mathrm{lc}(b_i')\}, W)$. This is done until every polynomial in the current basis F' has its leading coefficient specializing to not null.

From this point we proceed with Buchberger's algorithm: first we construct a list of pairs $\{(f_i, f_j) : f_i, f_j \in F'\}$, such that S-pol$(f_i, f_j)$ do not reduce to zero by Buchberger's criteria, sorted by the Normal Strategy [1]. Then we take the first pair from the list and compute its S-polynomial, S_{ij}, testing its leading coefficient as we have done for the polynomials in F', and then add the resulting S'_{ij} to the current basis F'. If a new branch condition is found, it stops.

When a Buchberger branch has finished, it marks the current vertex of the tree as final and the main recursive algorithm `CONDPGB` goes back to the next branch to be done.

[1]http://www-ma2.upc.es/~montes/

DisPGB20 algorithm

```
DISPGB($B, \succ_{\bar{x}}, \succ_{\bar{a}}$)
Input: $B \subseteq R[\bar{a}][\bar{x}]$,
$\succ_{\bar{x}}$, $\succ_{\bar{a}}$ termorders wrt the variables $\bar{x}$ and the parameters $\bar{a}$ respectively.
Output: $T$ a table with binary tree structure.
BEGIN
  global variable $T := \phi$;
  CONDPGB($B, \phi$);
END

CONDPGB($B, \Sigma$)
Input: $B \subseteq R[\bar{a}][\bar{x}]$, $\Sigma = (N, W)$ a semi-canonical specification.
BEGIN
  $cf := false$;
  $(cb, cd, B', \Sigma_0, \Sigma_1)$:=CONDTOBRANCH($B, \Sigma$);
  IF $cd$ THEN
    $(cb, cf, B', \Sigma_0, \Sigma_1)$:=CONDBUCHBERGER($B', \Sigma_1$);
  END IF
  Store data in T;
  IF $cf$ THEN
    Mark current vertex in $T$ as terminal.
    RETURN();
  ELSE
    IF $cb$ THEN
      CONDPGB($B', \Sigma_0$);
      CONDPGB($B', \Sigma_1$);
    ELSE
      CONDPGB($B', \Sigma_1$);
    END IF
  END IF
END
```

$(cb, cd, B', \Sigma_0, \Sigma_1) \leftarrow$ CONDTOBRANCH(B, Σ)
Input: $B \subseteq R[\bar{a}][\bar{x}]$, $\Sigma = (N, W)$ a semi-canonical specification.
Output: Reduce polynomials in B wrt N. If there is some polynomial $b_i \in B$ with its leading coefficient still not decided to not null wrt Σ, then build up two new semi-canonical specifications (Σ_0, Σ_1) extending the given Σ.
Σ_1 is obtained supposing the new condition is not null and Σ_0 is obtained supposing the new condition is null, if this assumption is compatible with the latest specification Σ. In this case return $cb = true$.
If all polynomials in B' have leading coefficient decided to not null, then return $cd = true$.

$(cb, cf, B', \Sigma_0, \Sigma_1) \leftarrow$ CONDBUCHBERGER(B, Σ)
Input: $B \subseteq R[\bar{a}][\bar{x}]$, $\Sigma = (N, W)$ a semi-canonical specification.
Output: Perform Buchberger algorithm with B. Each non-vanishing S-polynomial is reduced wrt Σ using `CONDTOBRANCH`.
If this `CONDTOBRANCH` returns $cb = false$ then the current `CONDBUCHBERGER` stops and returns $cb = false$ together with the curret values.
Whenever Buchberger algorithm finishes, then it returns $cf = true$.

2 Some test of DisPGB20

We made some other practical improvements on this algorithm, to make it run faster and not wasting much time in useless already done computations. In this respect, we improved the `GGE` routine to avoid divisions for which we already know the result before performing them. For the main algorithm presented above, whenever possible, we do not carry out tests already done to the basis and to the S-polynomials.

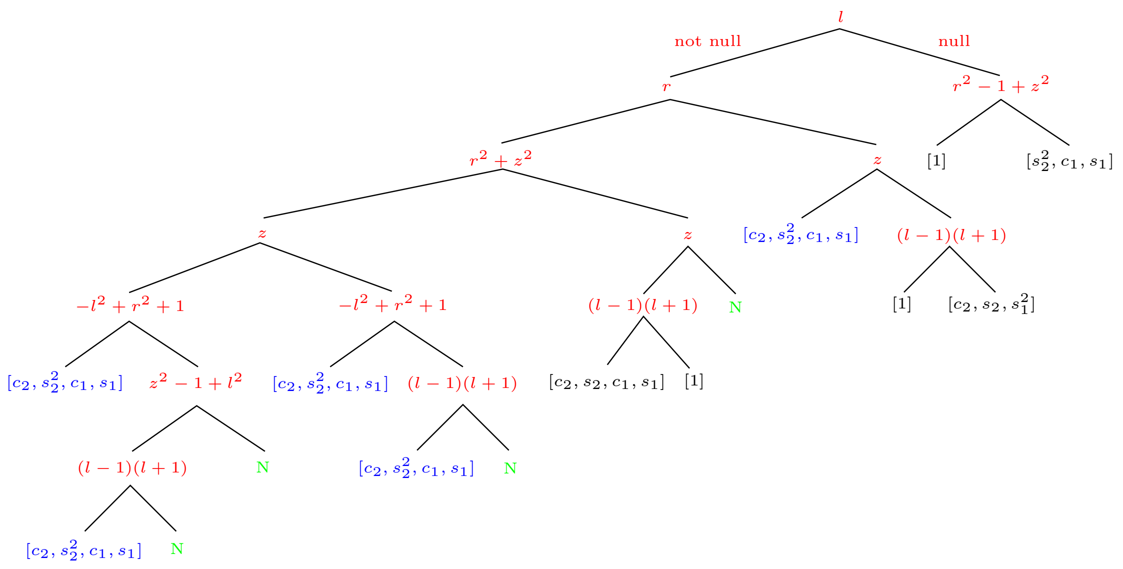

Figure 1: Output discussion for the simple robot problem, using release 1.4

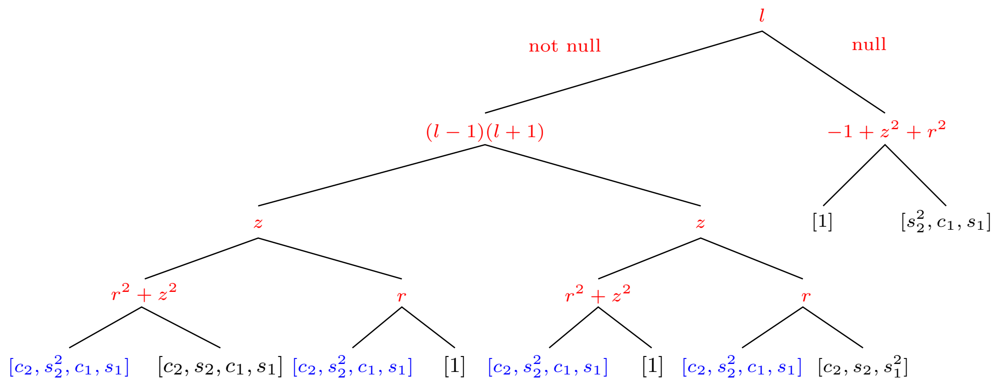

Figure 2: Output discussion for the simple robot problem, using release 2.0

For the choice of pairs in `CONDBUCHBERGER` we tried different strategies, namely Normal strategy [1], Sugar strategy and Double Sugar strategy [5]. The last two ones didn't work as well as we expected, so we decided for the Normal strategy. We did not try Faugère strategy [4], but we think it could be promising.

The following table compares times using releases 1.4 and 2.0. We can see the important increase of speed:

	release 1.4 time	release 2.0 time
Load flow problem	12.41 s	8.4 s
Simple robot example	31.93 s	4.1 s

Comparing outputs for the problem of the simple robot, release 2.0 provides a much simpler set of output bases and, as one can observe in figures 1 and 2, the output tree discussion is much better than with release 1.4.

References

[1] T. Becker and V. Weispfenning. Gröbner Bases: A Computational Approach to Commutative Algebra. *Graduate Texts in Mathematics*, 141, Springer Verlag, 1993.

[2] J. M. Brunat and A. Montes. The characteristic ideal of a finite, connected, regular graph. To be presented at *ISSAC*, 2004.

[3] D. Duval. Evaluation dynamique et clôture algébrique. *J. Pure and Applied Algebra*, **99**:267–295, 1995.

[4] J.-C. Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5). *Proc. ISSAC 2002.* ACM-Press, 75–83, 2002.

[5] A. Giovini, T. Mora, G. Niesi, L. Robbiano and C. Traverso. "One sugar cube, please" or Selection strategies in the Buchberger algorithm. *Proc. ISSAC 1991.* ACM-Press, 49–54, 1991.

[6] 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.

[7] A. Montes. New algorithm for discussing Gröbner bases with parameters. *J. Symbolic Comput.*, **33**(1-2):183–208, 2002.

[8] Y. Sato and A. Suzuki. An alternative approach to comprehensive Gröbner bases. *J. Symbolic Comput.*, **36**:649–667, 2003.

[9] V. Weispfenning. Comprehensive Gröbner Bases. *J. Symbolic Comput.*, **14**(1):1–30, 1992.

[10] V. Weispfenning. Canonical Comprehensive Gröbner bases. *Proc. ISSAC 2002.* ACM-Press, 270–276, 2002.

Montserrat Manubens & Antonio Montes,
Universitat Politècnica de Catalunya,
{Montserrat.Manubens, Antonio.Montes}@upc.es