PARALLELIZED COMPUTATION OF EXTENDED UNIVERSAL GRÖBNER BASIS

© **Dmitry Alekseevich Pavlov**
Saint-Petersburg State Polytechnical University, Polytechnicheskaya 29, St.-Petersburg, 195251, Russia, Post-graduate Student of Applied Mathematics Department,
e-mail: dmitry.pavlov@gmail.com

Key words: universal Gröbner basis; polynomial ideal; Young diagram.

The article presents an algorithm to calculate Extended Universal Gröbner Basis (EUGB), working on wide range of polynomial ideals. The EUGB($\mathfrak{A}$) of a polynomial ideal $\mathfrak{A}$ is defined as a finite [1] set of polynomials $\{f_i\}$ whose Young diagrams $Y(f_i)$ meet the following condition: $\dim(\mathcal{L}(Y(f_i))\cap\mathfrak{A}) = 1$ (where $\mathcal{L}$ denotes the span of a set of polynomials in the quotient algebra of the ideal). It is known that the EUGB contains the Universal Gröbner Basis. The algorithm is based on geometry of Young diagrams in $\mathbb{Z}_+^d$, and finds the polynomials of EUGB mostly independently, which makes it able to run in parallel. An outline of the parallel version of the algorithm is given.

1 Notation and Background

Let $K[x_1,\dots,x_d]$ be a polynomial ring over a field K in d variables $X = \{x_1,\dots,x_d\}$. The space of monomials in these variables can be trivially identified with the lattice $\mathbb{Z}_{\geqslant 0}^d$. Here and after, we make no difference between the monomials and integer vectors with nonnegative coordinates—the elements of the lattice.

A polynomial ideal [2], generated by polynomials $(f_1,\dots,f_s)$, is defined as the following infinite set of polynomials from $K[x_1,\dots,x_d]$:

$$\langle f_1,\dots f_s\rangle = \left\{\sum_{i=1}^{s} h_i f_i : h_1,\dots,h_s \in K[x_1,\dots,x_d]\right\}.$$

It is known that the ideal can have more than one possible set of generators, each of which is called a basis of the ideal [2].

Let $\succ$ be a total order on $\mathbb{Z}_{\geqslant 0}^d$. It is called admissible, when it meets the following condition:

- $\alpha \succ 0$ for each $\alpha \neq \mathbf{0}$;
- If $\alpha \succ \beta$, then $\alpha+\gamma \succ \beta+\gamma$ for each $\gamma \in \mathbb{Z}_{\geqslant 0}^d$.

We denote as $\mathrm{LT}_\succ(f)$ the *leading term* of the polynomial f, that is, the term whose monomial is the biggest according to the admissible ordering $\succ$.

Let $\mathfrak{A}$ be a polynomial ideal in $K[x_1,\dots,x_d]$ and $\succ$ be an admissible monomial ordering. A finite set of polynomials $G \in \mathfrak{A}$ is called a Gröbner basis $\mathrm{GB}_\succ(\mathfrak{A})$ of $\mathfrak{A}$, if the leading term of any polynomial from $\mathfrak{A}$ is a multiple of some leading term of a basis polynomial [2]:

$$\langle\{\mathrm{LT}_\succ(g_i) : g_i \in G\}\rangle = \langle\{\mathrm{LT}_\succ(f) : f \in \mathfrak{A}\}\rangle.$$

The Gröbner basis G of the ideal $\mathfrak{A}$ has two important properties: first, it generates $\mathfrak{A}$, and second, every polynomial $g \in \mathfrak{A}$ has a *normal form* r, defined as a result of polynomial reduction of g w.r.t. G with the monomial order $\succ$:

$$g = h_1 f_1 + \cdots + h_s f_s + r, \qquad h_i, r \in K[x_1, \ldots, x_d].$$

By the definition of polynomial reduction, no term of r is a multiple of any of $\mathrm{LT}_\succ(f_i)$ (*).

2 Coideals, quotient algebra, and FGLM algorithm

The nondivisibility condition (*) has a convenient geometrical interpretation: all monomials of r are positioned "under" the monomial ideal, formed by $\{\mathrm{LT}(f_i), f_i \in \mathrm{GB}_\succ(\mathfrak{A})\}$. That is, they belong to the *coideal* $\mathrm{Co}(GB_\succ(\mathfrak{A})) = \mathbb{Z}^d_{\geqslant 0} \backslash \langle\{\mathrm{LT}(f_i)\}\rangle$.

All possible normal forms of polynomials of $\mathfrak{A}$ w.r.t. $\mathrm{GB}_\succ(\mathfrak{A})$ belong to $\mathcal{L}(\mathrm{Co}(\mathrm{GB}_\succ(A)))$, where $\mathcal{L}$ denotes the span of a set of polynomials in the quotient algebra of the ideal. We denote it $Q_\succ(\mathfrak{A})$, as it actually determines the quotient algebra of the ideal $\mathfrak{A}$ (fig. 2):

$$Q_\succ(\mathfrak{A}) \sim K[x_1, \ldots, x_d]/\mathfrak{A}.$$

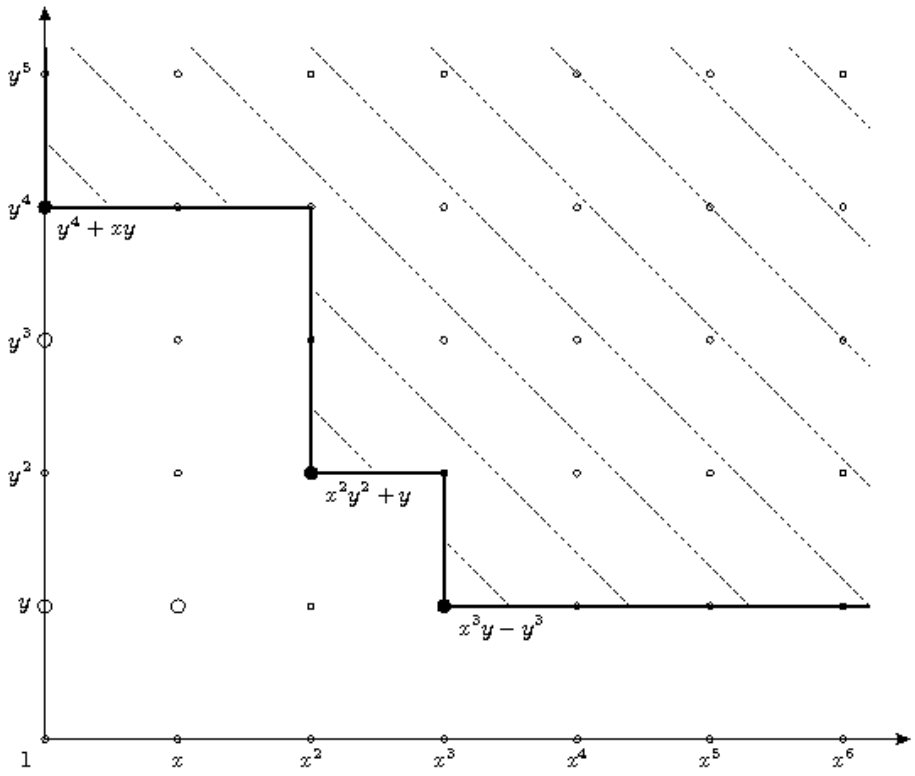

Fig. 1. The quotient algebra of an ideal generated by $\{y^4 + xy, x^2y^2 + y, x^3y - y^3\}$, is in turn generated by monomials, underlying the dashed area

We call monomials $\{m_i\}$ (not necessarily finite set) *linearly independent* w.r.t. $\mathfrak{A}$, if the intersection of their span with $\mathfrak{A}$ is also zero:

$$\mathcal{L}(\{m_i\} \cap \mathfrak{A}) = \{\mathbf{0}\}.$$

The ideal $\mathfrak{A}$ clearly do not contain any normal forms w.r.t. $\mathfrak{A}$, except the zero polynomial:

$$Q_{\succ}\mathfrak{A} \cap \mathfrak{A} = \{\mathbf{0}\}.$$

Hence, the monomials from the coideal $\mathrm{Co}(\mathrm{GB}_{\succ}(A))$ form a linearly independent set. But as soon as we add to this set the leading monomial of any polynomial of the Gröbner basis, we have this polynomial within the linear span of the set:

$$\mathcal{L}(Q_{\succ}(\mathfrak{A}) \cup \mathrm{LT}(f_i)) \cap \mathfrak{A} = \langle f_i \rangle,$$

and the dimension of this span is obviously equal to 1:

$$\dim(\mathcal{L}(Q_{\succ}(\mathfrak{A}) \cup \mathrm{LT}(f_i)) \cap \mathfrak{A}) = 1.$$

This way of Gröbner basis polynomials construction is used in FGLM [4] algorithm, which computes a Gröbner basis for an arbitrary monomial order $\succ'$, given another Gröbner basis for another monomial order $\succ$. Basically, the FGLM algorithms incrementally builds the coideal (starting from zero monomial), following the monomial order $\succ'$, until the monomials are not linearly independent. After they become linearly dependent, the corresponding polynomial of the Gröbner basis it calculated, and then the algorithms steps back and goes on, never adding that "linearly dependent" monomial again.

The FGLM algorithm has a limitation: it accepts only *zero-dimensional* polynomial ideals—the ideals whose quotient algebra is generated with a finite number of monomials. In another words, the coideal $\mathrm{Co}(\mathrm{GB}_{\succ}(A))$ in this case is finite, i.e. zero-dimensional.

3 Universal Gröbner basis and Young diagrams

We denote as $\mathrm{UGB}(\mathfrak{A})$ the *universal Gröbner basis* of the ideal $\mathfrak{A}$: the union of all possible Gröbner basiss with all admissible monomial orders. Robbiano [3] has shown that the UGB is always finite.

We define a d-dimensional Young diagram as a subset of $\mathbb{Z}^d_{\geqslant 0}$ lattice, with the only requirement that if it contains some monomial m, it must also contain all divisors of m.

We call a Young diagram of a polynomial $r \in \mathfrak{A}$ the one formed by the terms (monomials) of r, that is, a union of monomials of r and all their divisors (fig. 3).

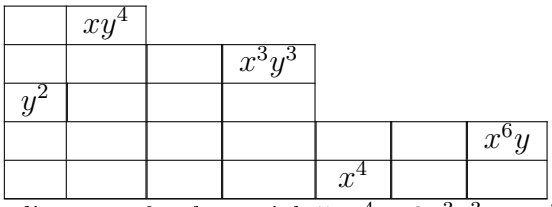

Fig. 2. Young diagram of polynomial $5xy^4 + 2x^3y^3 + xy^2 - x^6y + 8x^4$

For every polynomial $f \in \mathrm{UGB}(\mathfrak{A})$ the following condition is hold [1]:

$$\dim(\mathcal{L}(Y(f) \cap \mathfrak{A})) = 1. \tag{1}$$

This condition does not guarantee that the polynomial f is a part of $\mathrm{UGB}(\mathfrak{A})$; nevertheless, this condition is easier to check, as it does not imply any admissible monomial order.

We denote $\mathrm{EUGB}(\mathfrak{A})$ the *Extended Universal Gröbner basis* of the ideal $\mathfrak{A}$:

$$\mathrm{EUGB}(\mathfrak{A}) = \{f \in \mathfrak{A} : \dim(\mathcal{L}(Y(f) \cap \mathfrak{A})) = 1\}.$$

The $\mathrm{EUGB}(\mathfrak{A})$ is always finite [1]. Clearly, $\mathrm{UGB}(\mathfrak{A}) \subset \mathrm{EUGB}(\mathfrak{A})$. Unlike the finding of $\mathrm{UGB}(\mathfrak{A})$, the finding of $\mathrm{EUGB}(\mathfrak{A})$ is done via geometrical and combinatorial operation on $\mathbb{Z}^d_{\geqslant 0}$.

4 Finding EUGB($\mathfrak{A}$)

The described algorithm searches for Young diagrams whose spans have a one-dimensional intersection with the ideal $\mathfrak{A}$, and this intersection is itself an ideal generated by some polynomial from $\mathrm{EUGB}(\mathfrak{A})$.

The following global variables are used:

- `*basis*` — an arbitrary Gröbner basis to start with; for example, a Gröbner basis w.r.t. **degrevlex** monomial order. It is needed for checking the linear independence of the monomials of Young diagrams.
- `*diagrams*` — a list of found diagrams that fulfill the condition (1). Each diagram is defined by a list of nondivisor monomials (i.e. "corners" of the diagram. At the start, the list of diagrams contain the Young diagrams of the polynomials of the `*basis*`. (Although it could be empty, but then the procedure would have taken more time.)
- `*bad-coideals*` — a list of coideals that do not contain diagrams of interest of size less than `MAX-SIZE`. At the start, this list is empty.

The following helper functions are mentioned but not listed:

- `nondivisors (poly)`: accepts a polynomial and returns its monomials, that are not divisors of any other monomials of this polynomial.
- `contains-divisors-of (monomials, diagram)` finds among the `monomials` the divisors of `diagram`'s"corners".
- `remove-multiples-of (monomials, corner)` removes from `monomials` the ones that are multiples of `corner`.
- `intersect-with-ideal (sequence, basis)` checks the intersection of a span of the `sequence` and the ideal, generated by `basis`. If the intersection is not zero, it is assumed 1-dimensional, and the resulting generating polynomial is returned.
- `coideal-belongs (inner, outer)` checks that the `inner` monomial coideal is a subset of the `outer`.

- `closest-to-origin (list)` returns the monomial from the `list` that is closest to the origin.

As the first step, the algorithm outputs the polynomials of the given `*basis*`, which clearly meet the condition (1), and saves the diagrams of these polynomials. After that, the monomial coideals not containing the found `*diagrams*` are enumerated.

```
find-eugb (*basis*):
  *diagrams* ← {}
  for all poly ∈ *basis* do
    yield poly
    *diagrams* ← *diagrams* ∪ nondivisors(poly)
  end for
  repeat
    oldsize = size(*diagrams*)
    process-coideals(*diagrams*)
  until size(*diagrams*) = oldsize
```

In order to prevent duplicating diagrams in the output, each diagram is being searched for in a monomial coideal, which does *not* contain at least one "corner" of already found `*diagrams*`. Such a coideal (there can be many of them, but not infinitely many) is computed by the recursive procedure `process-coideals`. Once it is found, the `find-polynomial` function is invoked for this coideal.

```
process-coideals (diagrams, coideal = {}):
  if diagrams = {} then
    find-polynomial(coideal)
  else
    diagram ← any of the diagrams
    if contains-divisors-of(coideal, diagram) then
      process-coideals( diagrams\diagram , coideal)
    else
      for all corner ∈ diagram do
        new-coideal ← corner ∪ remove-multiples-of(coideal, corner)
        process-coideals( diagrams\diagram , new-coideal)
      end for
    end if
  end if
```

The next procedure accepts a coideal and grows a Young diagram inside it, starting from an empty diagram, and adding monomials one-by-one, until the condition 1 is met. On each step, from all monomials ("dimples") available for addition, the one closest to the origin is selected.

As the coideal may be infinite (especially in case we are dealing with a non-zero-dimension polynomial ideal), the procedure is forced to stop once the size of the coideal reaches `MAX-SIZE`.

```
find-polynomial (coideal):
  for all bad-coideal ∈ *bad-coideals* do
    if coideal-belongs(coideal, bad-coideal) then
      return
```

```
    end if
  end for
  seq ← {}
  dimples ← {0}
  while dimples ≠ {} do
    if |seq| > MAX-SIZE then
      print Linear-dependent Young diagram not found in coideal.
      *bad-coideals* = *bad-coideals* ∪ coideal
      return
    end if
    new-monom ← closest-to-origin(dimples)
    seq ← seq ∪ new-monom
    poly ← intersect-with-ideal(seq, *basis*)
    if poly ≠ 1 then
      *diagrams* ← *diagrams* ∪ support(poly)
      yield poly
    end if
    dimples ← update-dimples(new-monom, dimples\new-monom)
  end while
```

The helper procedure `update-dimples` adds a new "dimple" to the list of available monomials for the next step of diagram growing, and removes its divisors from the list.

```
update-dimples (dimples new-cell):
  for all v ∈ X do
    if ∄c ∈ dimples : c ≺ v · new-cell then
      dimples ← dimples ∪ v · new-cell
    end if
  end for
  return dimples
```

The above algorithm is able to find the polynomials of EUGB($\mathfrak{A}$), whose Young diagrams are of size less than `MAX-SIZE`. The size limit can be discarded for zero-dimensional ideals, where the endless growing of a diagram of linearly independent monomials is theoretically impossible. On all other ideals, the size limit allows to avoid endless loops, but can lead to loss of some EUGB polynomials with too big Young diagrams.

Unfortunately, there is no possibility to give any reasonable estimation for `MAX-SIZE` that would guarantee the generation of entire EUGB. Obviously, the size of a Young diagram of a polynomial can not be less than its degree; and the best known estimations on the degree of Gröbner basis elements are 2^{2^k} for **lex** ordering [5] and k^2+1 for **grevlex** ordering [6] (where k is the maximum degree amongst the generators of the ideal).

5 Parallelizing the EUGB finding algorithm

A lot of operations in the algorithm described above can be run in parallel. While `process-coideals` in the original algorithms outline is called step-by-step, with each next step having one more diagram, it is possible first to calculate a few coideals to search in, then search for EUGB polynomials in each coideal independently, using a separate working thread. The following statements should be considered when implementing the parallelized EUGB finding:

- It is possible that the parallel algorithm would find the same Young diagrams simultaneously in different computing threads. While this overhead is not likely to be completely eliminated, some techniques would help (see below).
- Each coideal given to the working thread should not contain any of the `*diagrams*` found so far. Also, the less monomials the "current" processed coideals have in common, the better.
- To further decrease the possibility of finding duplicate diagrams simultaneously, the following principle should be used when selecting the next monomial for diagram increment (in addition to `closest-to-origin`): once the monomial got into the diagram being built by a working thread, it is better not to add this monomial to a diagram being build in a neighbor working thread; this should be done only if no other options are left.
- Once all the working threads are done with their Young diagrams and return some polynomials, the control flow should go back to the main thread, where the results are stored, the duplicates are eliminated, and the new portion of monomial coideals is calculated for the next parallel computation step.

6 Acknowledgements

I wish to thank Nickolay Vasiliev for encouraging and supervising my work on this topic.

Список литературы

1. *Vasiliev N.* Monomial Orderings, Young Diagrams and Gröebner Bases // Proceedings of the International Conference "Computer Algebra in Scientific Computing" (CASC). Technical University of Munchen. Munchen, 2003.
2. *Cox D.A., Little J.B., O'Shea D.* Ideals, Varieties, and Algorithms // Introduction to Computational Algebraic Geometry and Commutative Algebra. New York: Springer, 2007.
3. *Robbiano L.* Term ordering on the polynomial ring // Proceedings of EUROCAL '85 (Linz), Lecture Notes in Computer Science. 1985. V. 204. P. 513-517.
4. *Faugère J. C., Gianni P., Lazard D., Mora T.* Efficient computation of zero-dimensional Gröbner bases by change of ordering // J. of Symbolic Computation. 1993. V. 16. Issue 4. P. 329-344.
5. *Mayr E., Meyer A.* The complexity of the word problem for commutative semigroups and polynomial ideals // Adv. Math. 1982. V. 46. P. 305-329.
6. *Lazard D.* Gröbner Bases, Gaussian elimination and resolution of systems of algebraic equations // Proceedings of the European Computer Algebra Conference on Computer Algebra (EUROCAL). London: Springer-Verlag, 1983. P. 146-156.

Accepted for publication 7.06.2010.

ПАРАЛЛЕЛЬНОЕ ВЫЧИСЛЕНИЕ РАСШИРЕННЫХ УНИВЕРСАЛЬНЫХ БАЗИСОВ ГРЁБНЕРА

© **Дмитрий Алексеевич Павлов**
Санкт-Петербургский государственный политехнический университет, Политехническая, 29, Санкт-Петербург, 195251, Россия, аспирант кафедры прикладной математики,
e-mail: dmitry.pavlov@gmail.com

Ключевые слова: универсальный базис Грёбнера; полиномиальный идеал; диаграмма Юнга.

В статье представлен алгоритм вычисления расширенного универсального базиса Грёбнера (EUGB), работающий на широком классе полиномиальных идеалов. EUGB($\mathfrak{A}$) полиномиального идеала $\mathfrak{A}$ определён как конечное [1] множество полиномов f_i, чьи диаграммы Юнга $Y(f_i)$ удовлетворяют следующему условию: $\dim(\mathcal{L}(Y(f_i)) \cap \mathfrak{A}) = 1$ (где $\mathcal{L}$ обозначает линейную оболочку множества полиномов в факторалгебре идеала). Известно, что EUGB содержит универсальный базис Грёбнера идеала. Алгоритм основан на геометрических свойствах диаграмм Юнга в $\mathbb{Z}_+^d$, и элементы EUGB находятся им по большей части независимо друг от друга, что позволяет вычислять их параллельно. В последней части статьи дана схема параллелизации алгоритма.